IMPLEMENTACIÓN DE UN MÉTODO DE PROCESAMIENTO DEL LENGUAJE NATURAL EN UN VOICEBOT SOBRE EL CULTIVO DE MAÍZ

SNEIDER ANDRES SANCHEZ ARAQUE
CRISTIAN CAMILO CUCUNUBÁ VIRACACHA

UNIVERSIDAD CATOLICA DE COLOMBIA

FACULTAD DE INGENIERÍA

PROGRAMA INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

BOGOTÁ, COLOMBIA

2019

Implementación de un método de procesamiento del lenguaje natural en un voicebot sobre el cultivo de maíz

Sneider Andres Sanchez Araque
Cristian Camilo Cucunubá Viracacha

Documento presentado como requisito para optar al título de:

Ingeniero de Sistemas

Director:

Juan Carlos Barrero Calixto

Línea de Investigación:

Machine Learning aplicado al procesamiento del lenguaje natural.

Grupo de Investigación:

Software inteligente y convergencia tecnológica

Universidad Catolica de Colombia

Facultad de ingenieria,Ingenieria de sistemas y computación

Bogotá, Colombia

2019

Atribución-NoComercial-SinDerivadas 2.5 Colombia (CC BY-NC-ND 2.5 CO)

Este es un resumen legible por humanos (y no un sustituto) de la licencia. Advertencia.

Usted es libre de:

Compartir — copiar y redistribuir el material en cualquier medio o formato

La licenciante no puede revocar estas libertades en tanto usted siga los términos de la licencia

Bajo los siguientes términos:

Atribución — Usted debe dar crédito de manera adecuada, brindar un enlace a la licencia, e indicar si se han realizado cambios. Puede hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que usted o su uso tienen el apoyo de la licenciante.

NoComercial — Usted no puede hacer uso del material con propósitos comerciales.

SinDerivadas — Si remezcla, transforma o crea a partir de el material, no podrá distribuir el material modificado.

NOTA DE ACEPTACIÓN

_____

_____

_____

_____

FIRMA DIRECTOR DEL PROYECTO

_____

FIRMA JURADO

_____

FIRMA JURADO

_____

Bogotá, D.C. Octubre 2019

AGRADECIMIENTOS

Gracias a cada una de las personas que fueron parte de este proyecto, por acompañarnos y aportar con sus valiosos consejos que se ven reflejados a lo largo de este proyecto y en nuestras vidas.

Al director Juan Carlos Barrero por su dedicación, aportes, asesoría y apoyo, que permitieron que este proyecto tuviera las bases científicas y la teoría necesaria para cumplir todos los objetivos y sortear los obstáculos a lo largo del camino .

A nuestras familias, por comprender los sacrificios necesarios y darnos la fuerza imprescindible para poder cumplir nuestras metas y más específicamente este proyecto.

A la Universidad Católica de Colombia, por brindarnos los espacios educativos para ampliar nuestros conocimientos y facilitar nuestro desarrollo como ingenieros, pero más importante, como personas.

CONTENIDO

LISTA DE IMÁGENES

Pág.

LISTA DE TABLAS

LISTA DE ANEXOS

LISTA DE ABREVIATURAS

API : Application Programming Interface

LUIS : Language Understanding Intelligent Service

NLU: Natural Language Understanding

PIB: Producto Interno Bruto

REST: Representational State Transfer

JSON: Javascript Object Notation

PLN: Procesamiento de Lenguaje Natural

DSL: Domain Specific Language

OWL: Web Ontology Language

NPM: Node Package Manager

VS: Visual Studio

Ej: Ejemplo

RESUMEN

En los últimos años se ha empezado a implementar en las empresas el uso de tecnologías como los chatbots y voicebots en diferentes campos como la medicina, el entretenimiento y la educación, gracias a los beneficios que ofrecen como la agilización de procesos, reducción de esfuerzo de trabajo y de disponibilidad.

En este proyecto se presenta un método que integra una ontología y una aplicación de procesamiento de lenguaje natural dentro de un voicebot, para resolver preguntas formuladas por agricultores de Colombia por medio de una llamada telefónica, ayudando a estas poblaciones en regiones de difícil acceso y con barreras tecnológicas a capacitarse en el área de la agricultura y el cultivo de maíz.

Para procesar el lenguaje natural se utilizó la aplicación comercial en la nube Watson Assistant, la ontología fue creada utilizando la metodología: Ontology Development 101 y el editor de ontologías Protégé. Como resultado de la implementación del método se obtuvo un voicebot capaz de responder preguntas basadas en el audiolibro “¡TU ASISTENTE A LA MANO! Manual del cultivo de maíz” por DEKALB”. Haciendo uso de una ontología, de la herramienta de NLU Watson, Elasticsearch como motor de búsqueda y Asterisk como gestor de llamadas, todos unidos por una API en Node.JS que respaldan la efectividad del modelo.

Abstract

In recent years, the use of technologies such as chatbots and voicebots in different fields such as medicine, entertainment and education has begun to be implemented in companies, thanks to the benefits they offer such as streamlining processes, reducing effort Work and availability. This project presents a method that integrates an ontology and a natural language processing application within a voicebot, to solve questions asked by Colombian farmers through a phone call, helping these populations in regions of difficult access and with technological barriers to be trained in the area of agriculture and corn cultivation. For the natural language processing a commercial application in the cloud will be used as Watson Assistant, the ontology will be created using the methodology: Development of ontology 101 and an ontology editor such as Protégé.

In recent years, the use of technologies such as chatbots and voicebots has begun to be implemented in different fields such as medicine, entertainment and education, thanks to the benefits they offer such as streamlining processes, reducing effort Work and availability.

This project presents a method that integrates an ontology and a natural language processing application within a voicebot, to solve questions asked by Colombian farmers through a phone call, helping these populations in regions of difficult access and with technological barriers to be trained in the area of agriculture and corn cultivation.

For natural language processing, was used the commercial application Watson Assistant cloud, the ontology was created using the methodology: "Development of ontology 101" and the editor of ontologies Protegé. As a result of the implementation of the method, a voicebot was obtained that is able to answer questions based on the audiobook "¡TU ASISTENTE A LA MANO! Manual del cultivo de maíz" por DEKALB". *Using an ontology, the NLU Watson tool, Elasticsearch as a search engine and Asterisk as a call manager, all linked by an API in Node.JS that support the solution of the model.*

Keywords: agricultura, análisis de datos, base de datos, inteligencia artificial, ontología.

INTRODUCCIÓN

Debido a la actual transformación digital, muchas empresas buscan mejorar su estrategia de negocio implementando soluciones tecnológicas innovadoras que brinden beneficios a sus usuarios y les permitan a estas empresas tener ventajas frente a sus competidores. “El avance de los últimos años en procesamiento del lenguaje natural, interfaces de conversación, automatización y procesos de *machine learning* y *deep learning*, ha permitido que los voicebots y asistentes virtuales sean cada vez más inteligentes y útiles” (Ibañez, 2019). Permitiéndoles ahorrar tiempo, esfuerzo y costos en tareas repetitivas como el servicio al cliente o en este caso, la respuesta a preguntas frecuentes acerca del cultivo de maíz.

Un *voicebot* es un software que interactúa con una persona por medio de lenguaje natural para poder ser capaz de responder preguntas correctamente. Grandes compañías han desarrollado múltiples *voicebot* o asistentes virtuales para crear soluciones industriales o de investigación, entre los más famosos se encuentran: Apple Siri, Microsoft Cortana, Facebook M y IBM Watson. (Clarizia 1 et al. 2019). Uno de los retos más importantes en la actualidad de los *voicebots* es la emulación de conversaciones naturales con un ser humano, esto comprende entender lo que la persona está diciendo y cómo el *voicebot* aplica procesamiento del lenguaje natural para extraer las piezas de información más relevantes y generar la respuesta correcta. (Altinok, 2019).

Esta práctica empresarial plantea implementar un *voicebot* en la empresa: IP Consultores S.A.S., que mejore la búsqueda de información, debido a que anteriormente la búsqueda realizaba manualmente como una contestadora, el proceso constaba de que el usuario escucha un menú y a partir de ello, digita el número del capítulo que contiene la información que busca y posteriormente sería reproducido. El *voicebot* fue implementado con un método que utiliza una aplicación comercial de procesamiento de lenguaje natural y una ontología para responder adecuadamente a las preguntas que generan los usuarios en el software.

1. GENERALIDADES

1.1 LÍNEA DE INVESTIGACIÓN

Campo de investigación: *Machine Learning* aplicado al análisis de lenguaje natural.

Grupo de Investigación: Software inteligente y convergencia tecnológica porque desarrolla proyectos en áreas del conocimiento como la Ciencia, Tecnología e Innovación para poder investigar y desarrollar componentes de software que integren soluciones para educación y pequeña empresa maximizando la producción y optimización de recursos económicos y ambientales con énfasis en los agricultores.

1.2 PLANTEAMIENTO DEL PROBLEMA

Una empresa del sector agroquímico con operaciones en Colombia, está interesada en brindar a sus clientes actuales y potenciales, en regiones agrícolas con barreras tecnológicas y de difícil acceso, nuevas fuentes de conocimiento que les permitan una formación autodidacta en temas agropecuarios tomando como fuente audiolibros en español que ellos mismos crean. El audiolibro tiene que ser distribuido a través de mecanismos tradicionales que no impliquen el uso de redes actuales de conexión a internet (como 4G/3G), dado la ubicación geográfica.

Junto al acceso de la tierra, la educación es posiblemente una de las apuestas más importantes que tendrá que hacer el país en los próximos años si quiere cerrar las brechas de desigualdad en el sector rural. Según las cifras de la institución, de aproximadamente 18.000 estudiantes apenas 152 provienen de las áreas rurales, quienes antes de escoger una ingeniería agronómica, pesquera o ambiental prefieren irse por la medicina, las ingenierías, la administración de empresas y el derecho. (Mercado 2019)

Es por eso que proyectos como estos, aumenta el interés y la participación de la población rural en áreas de conocimiento del campo. También parte del desinterés y la poca participación, se debe a la dificultad de acceso a las TIC como herramientas de aprendizaje autónomo y digital.

De acuerdo con el Ministerio de Tecnologías de la Información y Comunicaciones, en Colombia hay 20 millones de colombianos que no cuentan con internet de banda ancha. La gran mayoría están ubicados en zonas rurales o dispersas, lo que afecta la integración de nuevas tecnologías al sector agropecuario. Al ver la distribución por departamentos se puede notar que los territorios que basan su economía en actividades agrícolas tienen bajo acceso de internet fijo, en cambio, zonas industriales y con mayor población tienen un índice más elevado. Por ejemplo, el mayor índice de acceso de Internet fijo a nivel departamental, contando el Distrito Capital, lo lideró Bogotá, con 22,7%; seguido por los departamentos de Risaralda (17,3%) y Antioquia (17,2%). En cambio, departamentos como Sucre (5,8%), Córdoba (5,3%), Nariño (5,4%), Cauca (5,4%), Vichada (1,2%) o Vaupés (0,1%) tienen bajo acceso. (Ministerio de las TIC, 2019)

Previamente dentro de la empresa donde se trabajó, se implementó una contestadora tradicional para responder a las llamadas de las personas y así convertir el audiolibro en un menú de mensajes automáticos que se reproducen mediante la interacción del usuario con los números del teléfono, pero esta solución ofrece respuestas muy limitadas y nada flexibles.

Así que se construyó un *voicebot* básico que le pide directamente a la persona las palabras clave del tema que quiere consultar, ejemplo: [‘cultivar’, ‘maíz’] para armar una consulta a la base de datos con esas palabras y encontrar más fácil el capítulo del audiolibro que necesita escuchar la persona. Sin embargo, de esta forma el voicebot no puede recibir oraciones completas, porque no sabría cómo analizarlas.

Para que un *voicebot* pueda recibir preguntas completas y sea capaz de entenderlas, necesita realizar el procesamiento del lenguaje natural y entender aspectos fundamentales de la pregunta cómo las intenciones, las entidades, las palabras claves, entre otras. De esta forma se garantiza una interacción más natural con el usuario y la respuesta que obtendrá se espera sea la correcta.

1.2.1 Pregunta de Investigación

¿De qué manera se puede implementar la búsqueda de información en un audiolibro sobre el cultivo de maíz, usando procesamiento de lenguaje natural en un *voicebot*?

1.2.2 Alcances y limitaciones

El presente proyecto tiene planeado desarrollarse en el país de Colombia, haciendo hincapié en los campesinos del territorio, aunque cabe aclarar que cualquier ciudadano podrá utilizar los servicios del *voicebot* por medio de una llamada celular siempre y cuando se encuentre en el territorio colombiano. El servicio podrá ser utilizado por un promedio de 2.000 personas al tiempo y se tendrá un límite máximo de 20.000 peticiones al software elegido al mes.
Por otra parte, sólo se realizará la ontología para el audiolibro de agricultura entregado por la empresa interesada, respecto al proyecto, se va a realizar todo el experimento de implementación del software, teniendo en cuenta que para etapas futuras del proyecto se va desarrollar un software comercial, aplicando un ciclo de vida del software.

- El presente proyecto tiene planeado desarrollarse en el país de Colombia, haciendo hincapié en los campesinos del territorio, aunque cabe aclarar que cualquier ciudadano podrá utilizar los servicios del *voicebot* por medio de una llamada celular siempre y cuando se encuentre en el territorio colombiano. El servicio podrá ser utilizado por un promedio de 2.000 personas al tiempo y se tendrá un límite máximo de 20.000 peticiones al software elegido al mes.
- Por otra parte, sólo se realizará la ontología para el audiolibro de agricultura entregado por la empresa interesada, respecto al proyecto, se va a realizar todo el experimento de implementación del software, teniendo en cuenta que para etapas futuras del proyecto se va desarrollar un software comercial, aplicando un ciclo de vida del software.

1.3 JUSTIFICACIÓN

En principio, se plantea la necesidad de mejorar la búsqueda de información en un audiolibro, debido a que inicialmente se propuso la búsqueda como una tarea estática, en la cual el usuario únicamente podría consultar secciones o capítulos específicos oprimiendo un número de la contestadora, esto produce una “ruta” que conducirá al usuario hacia la información en la que está interesado. El

principal problema al ser una tarea estática es que lleva a la consulta a ser un proceso que no es efectivo ni rápido, debido a que la persona se demora mucho tiempo en encontrar el capítulo que contiene la información que desea consultar, además al realizar la búsqueda no se asegura que la información exista en el audiolibro, lo cual puede hacer perder el tiempo del usuario, también se debe tener en cuenta que el usuario puede elegir una "ruta" incorrecta lo cual hace poco eficaz la búsqueda.

Sin embargo, al simplificar el proceso de búsqueda para que la única interacción que realice el usuario, sea realizar una pregunta sobre el tema el cual quiere consultar y automáticamente el voicebot entienda su pregunta, mediante el uso de una herramienta de procesamiento de lenguaje natural apoyado por una ontología sobre el dominio de la información, para buscar en la base de datos el clip del audiolibro correcto y reproducirlo de vuelta a el usuario, ofrece ventajas como: disminución de tiempo de la llamada, satisfacción del cliente, disponibilidad a cualquier hora, automatización de tareas, entre otras.

Agregando a lo anterior, este proyecto también tiene un impacto a nivel latinoamericano, debido a que como se muestra en La *tabla .1 Proporción de la agricultura en el PIB nacional,* se observa que la agricultura tiene un gran papel en el PIB de los países latinoamericanos y del caribe. Por lo cual en trabajos futuros, este software podrá ser de gran ayuda no solo a los agricultores nacionales sino también en latinoamérica y el caribe en resolver las cuestiones que presenten respecto a sus cultivos.

Tabla 1. Proporción de la agricultura en el PIB nacional

Proporción de la agricultura en el PIB nacional

Países	1990	1995	2000	2005	2010	2014	Variación 1990-2010	Variación 1990-2014
Argentina	8,1	5,8	5,1	8,4	8,2	8,2	1%	1%
Barbados	3,8	3,5	2,3	1,8	1,7	-	-57%	-
Belice	20,0	17,8	16,8	15,4	13,2	-	-34%	-
Bolivia	16,7	16,9	15,0	14,4	12,9	-	-23%	-
Brasil	8,1	5,8	5,5	5,5	4,9	5,6	-40%	-31%
Chile	8,7	9,2	5,9	4,6	3,5	3,3	-60%	-62%
Colombia	16,7	15,3	8,9	8,4	7,1	6,7	-58%	-60%
Costa Rica	12,3	13,7	9,5	9,0	7,2	-	-42%	-
Cuba	14,0	8,8	8,4	5,6	5,0	-	-64%	-
Ecuador	21,4	22,6	16,3	10,0	10,2	9,4	-52%	-56%
El Salvador	17,4	14,5	10,5	10,6	12,6	-	-28%	-
Guatemala	-	-	-	13,4	11,8	11,5	-	-
Guyana	38,1	41,2	31,1	34,6	18,3	18,3	-52%	-52%
Haití	-	-	-	-	-	-	-	-
Honduras	22,4	21,5	15,9	13,7	12,5	13,8	-44%	-38%
Jamaica	-	9,0	7,0	5,9	6,3	-	-	-
México	7,8	4,4	3,5	3,4	3,5	3,5	-56%	-55%
Nicaragua	-	22,0	19,5	17,7	18,8	20,5	-	-
Panamá	9,8	7,9	7,2	7,0	3,8	-	-62%	-
Paraguay	-	20,1	15,8	19,6	22,5	20,9	-	-
Perú	-	9,3	9,0	7,8	7,2	-	-	-
República Dominicana	14,5	10,0	7,2	7,5	6,5	6,2	-55%	-57%
San Vicente y las Granadinas	21,2	14,1	8,6	6,3	7,2	7,7	-66%	-63%
Suriname	8,7	14,9	11,2	5,4	8,0	-	-8%	-
Trinidad y Tobago	2,6	2,4	1,4	0,5	0,7	-	-75%	-
Uruguay	9,2	8,6	7,0	10,4	8,8	8,6	-4%	-6%
Venezuela	5,5	5,5	4,2	4,0	5,8	-	6%	-

(Escobar 2019)

De igual manera se encontró que este software ayudará a el país a disminuir las brechas de conocimiento entre sus pobladores, apoyando procesos como Milagro y Agromatic pero a una escala nacional, debido a que el acceso es masivo por medio de la llamada telefónica, Además en futuros proyectos se podrían agregar muchas más fuentes de información para que el software sea capaz de resolver interrogantes de más áreas de la agricultura o de otro tema específico.

Como estudiantes se ve una oportunidad de aplicar los conocimientos adquiridos en este proyecto aportando a la sociedad, en especial a los campesinos en los procesos de formación y resolución de dudas en sus cultivos de maíz, teniendo en cuenta las limitaciones tecnológicas presentes.

1.4 OBJETIVOS

1.4.1 Objetivo General

- Implementar un método que integre una ontología con una aplicación comercial de procesamiento del lenguaje natural, dentro de un *voicebot*, para resolver preguntas a los usuarios con información de un audiolibro sobre el cultivo de maíz.

1.4.2 Objetivos específicos

- Realizar un análisis comparativo entre herramientas comerciales de procesamiento del lenguaje natural para implementar la más adecuada en la infraestructura actual del *voicebot*.
- Construir una ontología, basada en el contenido del audiolibro en el lenguaje natural del usuario, para ser utilizada en el entrenamiento de la herramienta de PLN.
- Construir un método a partir de la ontología y de la herramienta NLP escogida, para desarrollar un modelo que clasifique las intenciones en las preguntas hechas por los usuarios.
- Validar el método usando métricas de desempeño usando precisión. Recall para medir la calidad del modelo.

1.5. MARCOS DE REFERENCIA

1.5.1 Marco Conceptual

1.5.1.1 Comprensión de lenguaje natural

La comprensión del lenguaje natural (NLU, por sus siglas en inglés) es una rama de la inteligencia artificial (AI) que utiliza programas informáticos para comprender las entradas hechas en forma de oraciones en formato de texto o habla. (Rouse 2019) NLU utiliza algoritmos para reducir el habla humana a una ontología estructurada.
El objetivo principal detrás de NLU es crear bots habilitados para chat y voz que puedan interactuar de manera efectiva con el público sin supervisión.

1.5.1.2 Ontología

Es el modelo de trabajo de entidades e interacciones en algún dominio particular del conocimiento o las prácticas, Según Tom Gruber un especialista en inteligencia artificial de la Universidad de Stanford, una ontología es, "la especificación de conceptualizaciones, utilizada para ayudar a los programas y a los seres humanos a compartir el conocimiento". En este uso, una ontología es un conjunto de conceptos, como cosas, eventos y relaciones, que se especifican de alguna manera (como un lenguaje natural específico) para crear un vocabulario acordado para el intercambio de información. (Rouse 2019)

1.5.1.3 Lenguaje Natural

En computación, el lenguaje natural se refiere a un lenguaje humano como el inglés, el ruso, el alemán o el japonés, a diferencia de lenguajes de programación o comandos por consola con los que se habla generalmente a una computadora. El término generalmente se refiere a un idioma escrito, pero también puede aplicarse al lenguaje hablado. (Rouse 2019)

1.5.1.4 IBM Watson

Watson es una supercomputadora de IBM que combina inteligencia artificial (AI) y un sofisticado software analítico para tener un rendimiento óptimo como una máquina de "respuesta a preguntas". (Rouse 2019)

1.5.1.5 IBM Watson Knowledge Studio

IBM Watson Knowledge Studio es una aplicación en la nube que permite a los desarrolladores y expertos en dominios colaborar en la creación de componentes de anotadores personalizados que se pueden usar para identificar menciones y relaciones en texto no estructurado. (Sinha 2019)

1.5.1.6 IBM Watson *Natural Language Understanding*

IBM Watson *Natural Language Understanding* analiza el texto para extraer metadatos del contenido, como, por ejemplo, entidades, palabras clave, categorías, sentimientos, emociones, relaciones y roles semánticos. Con los modelos de anotación personalizados desarrollados con *Watson Knowledge studio*, se puede adaptar una solución a entidades y relaciones específicas de la industria o el dominio en texto no estructurado. («Watson Natural Language Understanding - Overview - Philippines» 2019)

1.5.1.7 Software de reconocimiento de voz

Es una herramienta que analiza el sonido hablado por la persona e intenta convertirlo a texto, utilizando redes neuronales que funcionan como nuestros cerebros para reconocer la frecuencia y acento del habla humano. Los software de reconocimiento de voz se usa comúnmente para operar un dispositivo, ejecutar comandos o escribir sin tener que usar un teclado, mouse o presionar cualquier botón. (Hope 2019)

1.5.1.8 Protégé

Protégé es un editor de ontologías de código abierto y un marco de base de conocimiento gratuitos. Protégé es apoyada por una fuerte comunidad de académicos, Gobierno, y usuarios corporativos, que usan Protégé para construir Soluciones basadas en el conocimiento en áreas tan diversas como la biomedicina, Comercio electrónico, y modelación organizacional. (Papacchini 2019)

1.5.1.9 ElasticSearch

Elasticsearch es un servidor de base de datos de fuente abierta, independiente desarrollado en Java. Principalmente, se utiliza para la búsqueda y análisis de texto completo.(Chand 2019) Toma datos no estructurados de varias fuentes y los almacena en un formato sofisticado que está altamente optimizado para búsquedas basadas en el idioma. Cada servicio de elasticsearch(Index API, Get API, Search API, Put Mapping API) está expuesto como una REST API que provee los siguientes características (Rouse 2019):

- Proporciona una solución de búsqueda escalable.
- Realiza búsquedas casi en tiempo real.
- Agiliza los procesos de copia de seguridad y garantiza la integridad de los datos.
- Un índice se puede recuperar fácilmente en un caso de una falla del servidor.
- Utiliza la notación de objetos Javascript (JSON) y las interfaces del programa de aplicación Java (API).
- Automáticamente indexa documentos JSON.
- Cada índice puede tener su propia configuración.

1.5.1.10 JavaScript

JavaScript (JS) es un lenguaje ligero e interpretado, orientado a objetos con funciones de primera clase, más conocido como el lenguaje de script para páginas web, pero también usado en muchos entornos sin navegador, tales como node.js, Apache CouchDB y Adobe Acrobat. («JavaScript» 2019) Es un lenguaje

de programación dinámico que soporta construcción de objetos basado en prototipos. La sintaxis básica es similar a Java y C++ con la intención de reducir el número de nuevos conceptos necesarios para aprender el lenguaje. Las construcciones del lenguaje, tales como sentencias if, y bucles for y while, y bloques switch y try ... catch funcionan de la misma manera que en estos lenguajes (o casi). («Acerca de JavaScript» 2019)

1.5.1.11 Node JS

Node.js es un entorno de ejecución de JavaScript de código abierto y multiplataforma. Es una herramienta de desarrollo popular para casi cualquier tipo de proyecto. Node.js ejecuta el motor de JavaScript V8, el núcleo de Google Chrome, fuera del navegador. Esto permite que Node.js tenga un gran rendimiento. (Collina and Copes 2019)

Node.js puede utilizarse para crear diferentes tipos de aplicaciones, como una aplicación de línea de comandos, aplicaciones web, aplicaciones de chat en tiempo real, servidores de API REST, entre otros. Sin embargo, se utiliza principalmente para crear programas de red como servidores web, similares a PHP, Java, o ASP.NET. (¿«What is Node.js?» 2019)

1.5.1.12 Procesamiento de lenguaje natural

El procesamiento del lenguaje natural (PNL) es la capacidad de un programa de computadora para comprender el lenguaje humano tal como se habla.
El desafío con el desarrollo de aplicaciones de PNL es más complicado porque las computadoras tradicionalmente requieren que los humanos "hablen" con ellas en un lenguaje de programación que sea preciso y altamente estructurado, o mediante un número limitado de comandos de voz claramente enunciados. Sin embargo, el habla humana no siempre es precisa: a menudo es ambigua y la estructura lingüística puede depender de muchas variables complejas, como la jerga, los dialectos regionales y el contexto social.(Rouse 2019)

1.5.1.13 Express

Es el *framework* web más popular de Node, y es la librería subyacente para un gran número de otros *frameworks* web de Node populares. Proporciona mecanismos para:

- Escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL (rutas).
- Integración con motores de renderización de "vistas" para generar respuestas mediante la introducción de datos en plantillas.
- Establecer ajustes de aplicaciones web como qué puerto usar para conectar, y la localización de las plantillas que se utilizan para renderizar la respuesta.
- Añadir procesamiento de peticiones "middleware" adicional en cualquier punto dentro de la tubería de manejo de la petición.

(MDN contributors 2019)

1.5.1.14 Asterisk

Asterisk es un *framework* de código abierto para crear aplicaciones de comunicaciones. Asterisk convierte una computadora ordinaria en un servidor de comunicaciones. Asterisk alimenta sistemas *IP PBX*, puertas de enlace *VoIP*, servidores de conferencia y otras soluciones personalizadas. Es utilizado por pequeñas empresas, grandes empresas, centros de llamadas, operadores y agencias gubernamentales de todo el mundo. Asterisk es gratis y de código abierto. Asterisk está patrocinado por Digium. (Digium 2019)

1.5.1.15 Github

Es un sitio web y un servicio en la nube que ayuda a los desarrolladores a almacenar y administrar su código, al igual que llevar un registro y control de cualquier cambio sobre este código. (Kinsta 2019)

1.5.1.16 Git

Git es un sistema de control de versiones distribuido, diseñado y desarrollado inicialmente por Linus Torvalds en el 2005 cuando BitKeeper, el sistema de

control de versiones que utilizaban para el desarrollo de Linux, cambiará su licencia y no permitiera su uso libre. (Escobar 2019)

1.5.1.17 NPM

De sus siglas NPM (Node Package Manager) es un gestor de paquetes desarrollado en su totalidad bajo el lenguaje JavaScript por Isaac Schlueter, a través del cual podemos obtener cualquier librería con tan solo una sencilla línea de código, lo cual nos permitirá agregar dependencias de forma simple, distribuir paquetes y administrar eficazmente tanto los módulos como el proyecto a desarrollar en general. (Muradas 2019)

1.5.1.18 Terminal

Se define como Terminal, aunque también es conocido bajo el nombre de Consola, a todo dispositivo electrónico que forma parte del Hardware de un ordenador, y que tiene la funcionalidad básica de ingresar o mostrar los datos que se encuentran dentro de una computadora o en un determinado sistema de computación. («Terminal» 2019)

1.5.1.19 Visual Studio Code

Es un editor de código ligero pero potente que se ejecuta en su escritorio y está disponible para Windows, macOS y Linux. Viene con soporte incorporado para JavaScript, TypeScript y Node.js y tiene un amplio ecosistema de extensiones para otros lenguajes (como C ++, C #, Java, Python, PHP, Go) y tiempos de ejecución (como .NET y Unity). (Microsoft 2019)

1.5.1.20 Google Forms

Google Forms (Formularios de Google), nos permite crear un simple formulario dependiendo de las necesidades que tengamos a su vez nos facilita el trabajo de tabulación ya que al realizar estas encuestas de manera online los datos que se ingresan son almacenados en una hoja de cálculo lo cual no ayuda con el trabajo con los datos obtenidos. (Núñez 2019)

1.5.2 Marco teórico

1.5.2.1 *Machine Learning*

Machine Learning es una disciplina científica del ámbito de la Inteligencia Artificial que crea sistemas que aprenden automáticamente. Aprender en este contexto quiere decir identificar patrones complejos en millones de datos. La máquina que realmente aprende es un algoritmo que revisa los datos y es capaz de predecir comportamientos futuros. Automáticamente, también en este contexto, implica que estos sistemas se mejoran de forma autónoma con el tiempo, sin intervención humana. (González 2019).

1.5.2.2 *Deep Learning*

El *deep learning* es un tipo de *machine learning* que entrena a una computadora para que realice tareas como las hacemos los seres humanos, como el reconocimiento del habla, la identificación de imágenes o hacer predicciones. En lugar de organizar datos para que se ejecuten a través de ecuaciones predefinidas, el *deep learning* configura parámetros básicos acerca de los datos y entrena a la computadora para que aprenda por cuenta propia reconociendo patrones mediante el uso de muchas capas de procesamiento. (SAS 2019)

1.5.2.3 Matriz de confusión o error

“Confusión o *error Matrix* es una tabla que describe el rendimiento de un modelo supervisado de Machine Learning en los datos de prueba, donde se desconocen los verdaderos valores. Se llama “matriz de confusión” porque hace que sea fácil detectar dónde el sistema está confundiendo dos clases.

1. ***True Positives* (TP):** cuando la clase real del punto de datos era 1 (Verdadero) y la predicha es también 1 (Verdadero)
2. **Verdaderos Negativos (TN):** cuando la clase real del punto de datos fue 0 (Falso) y el pronosticado también es 0 (Falso).
3. ***False Positives* (FP):** cuando la clase real del punto de datos era 0 (False) y el pronosticado es 1 (True).
4. ***False Negatives* (FN):** Cuando la clase real del punto de datos era 1 (Verdadero) y el valor predicho es 0 (Falso).” (sitio big data 2019)

1.5.2.4 *Machine Learning*: Precisión

Se refiere a la dispersión del conjunto de valores obtenidos a partir de mediciones repetidas de una magnitud. Cuanto menor es la dispersión mayor la precisión. Se representa por la proporción entre el número de predicciones correctas (tanto positivas como negativas) y el total de predicciones. (sitio big data 2019)

1.5.2.5 *Machine Learning:* Recall

Es el número de elementos identificados correctamente como positivos del total de positivos verdaderos. (sitio big data 2019)

1.5.2.6 Puntuación F1

F1 es una medida general de la precisión de un modelo que combina precisión y recuperación, de esa forma extraña que la suma y la multiplicación simplemente mezclan dos ingredientes para hacer un plato por completo. Es decir, un buen puntaje F1 significa que tiene bajos falsos positivos y bajos falsos negativos, por lo que está identificando correctamente las amenazas reales y no le molestan las falsas alarmas. Un puntaje F1 se considera perfecto cuando es 1, mientras que el modelo es un fracaso total cuando es 0. (sitio big data 2019)

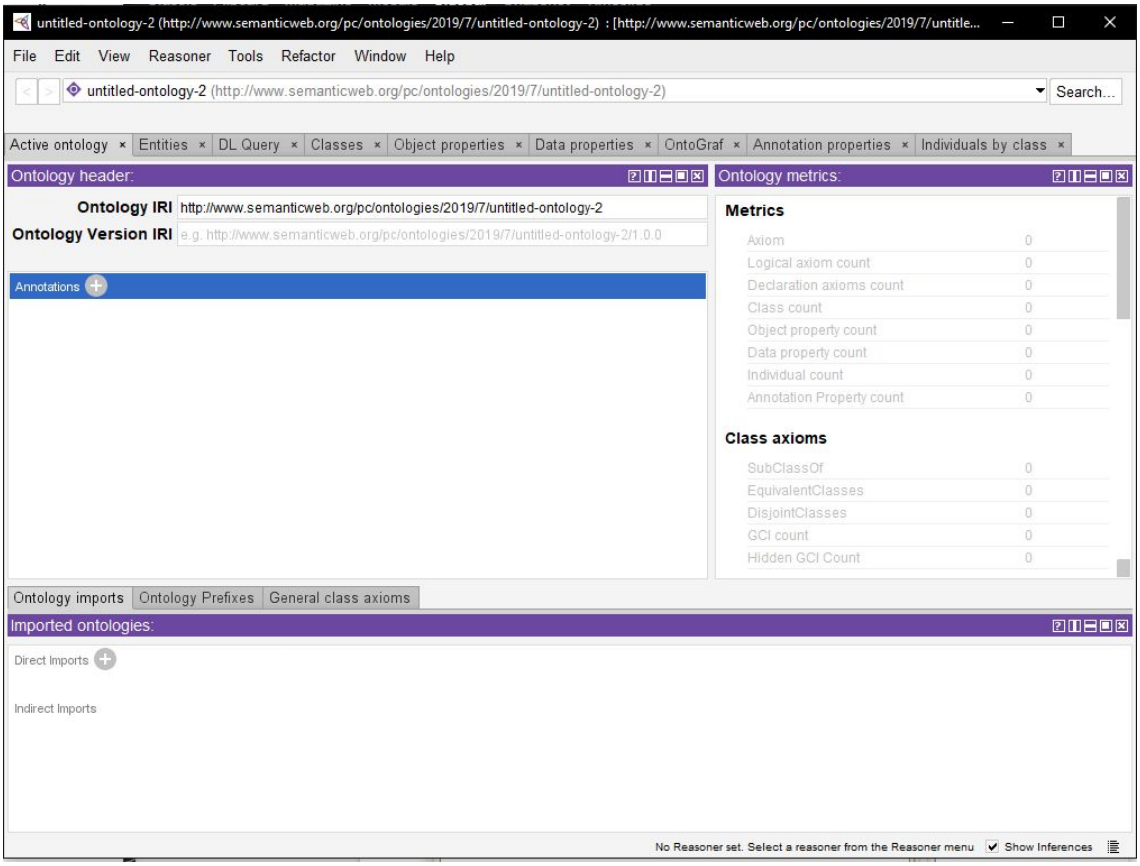
1.5.2.7 Protégé

Protégé es una plataforma gratuita de código abierto que proporciona a una comunidad de usuarios en crecimiento un conjunto de herramientas para construir modelos de dominio y aplicaciones basadas en el conocimiento con ontologías. (Research 2019)

Para su instalación se tienen dos opciones disponibles, vía web y a través de un programa ejecutable de Windows, las funcionalidades adquiridas por cualquiera de las dos opciones son casi idénticas con la diferencia de que el entorno web está más orientado a realizar ontologías en grupo, soportando un control de cambios y un historial en múltiples usuarios que pueden colaborar para realizar la ontología, cabe aclarar que las versiones son compatibles entre sí. (Research 2019)

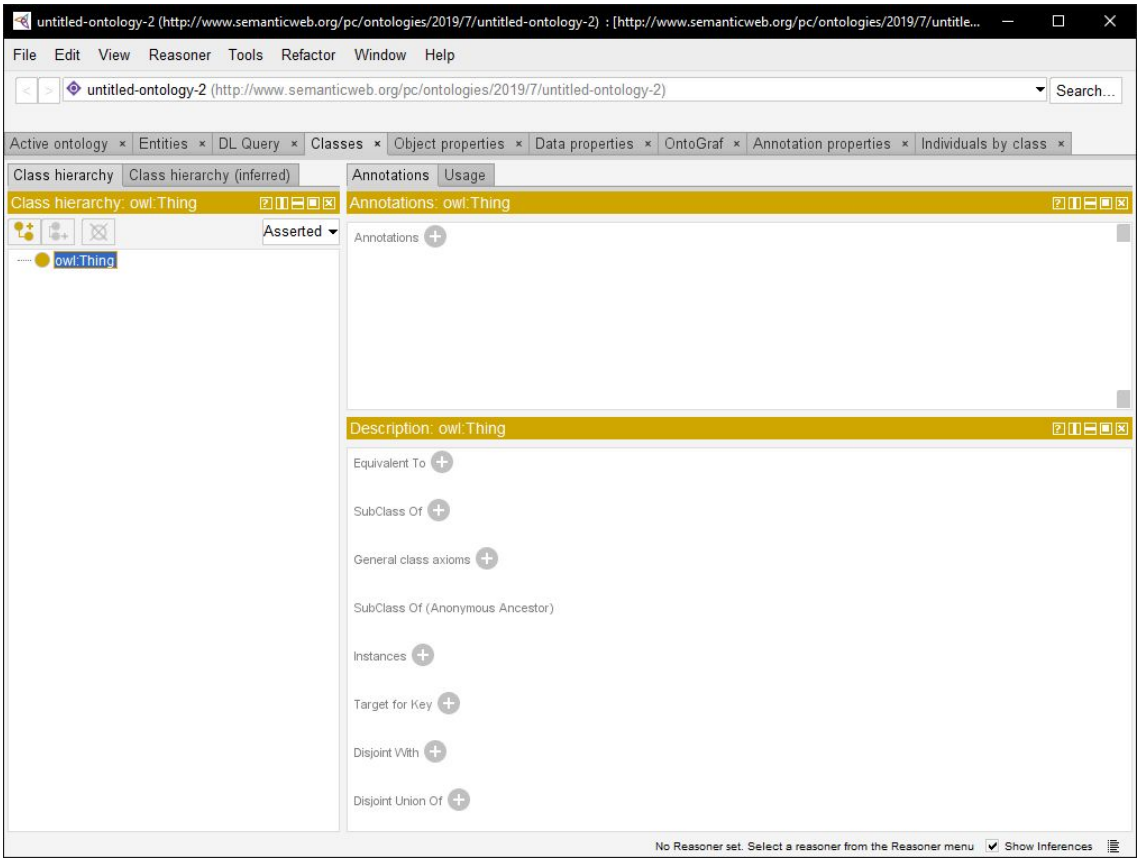
El software tiene una interfaz muy amplia, con múltiples opciones, para mayor información sobre las capacidades del software puede visitar "https://protege.stanford.edu/support.php#documentationSupport" donde encontrará documentación detallada del software. En la *imagen 1. Interfaz de* Protégé encontrará un pantallazo de la vista del dashboard de Protégé.

Imagen 1. Interfaz de protege



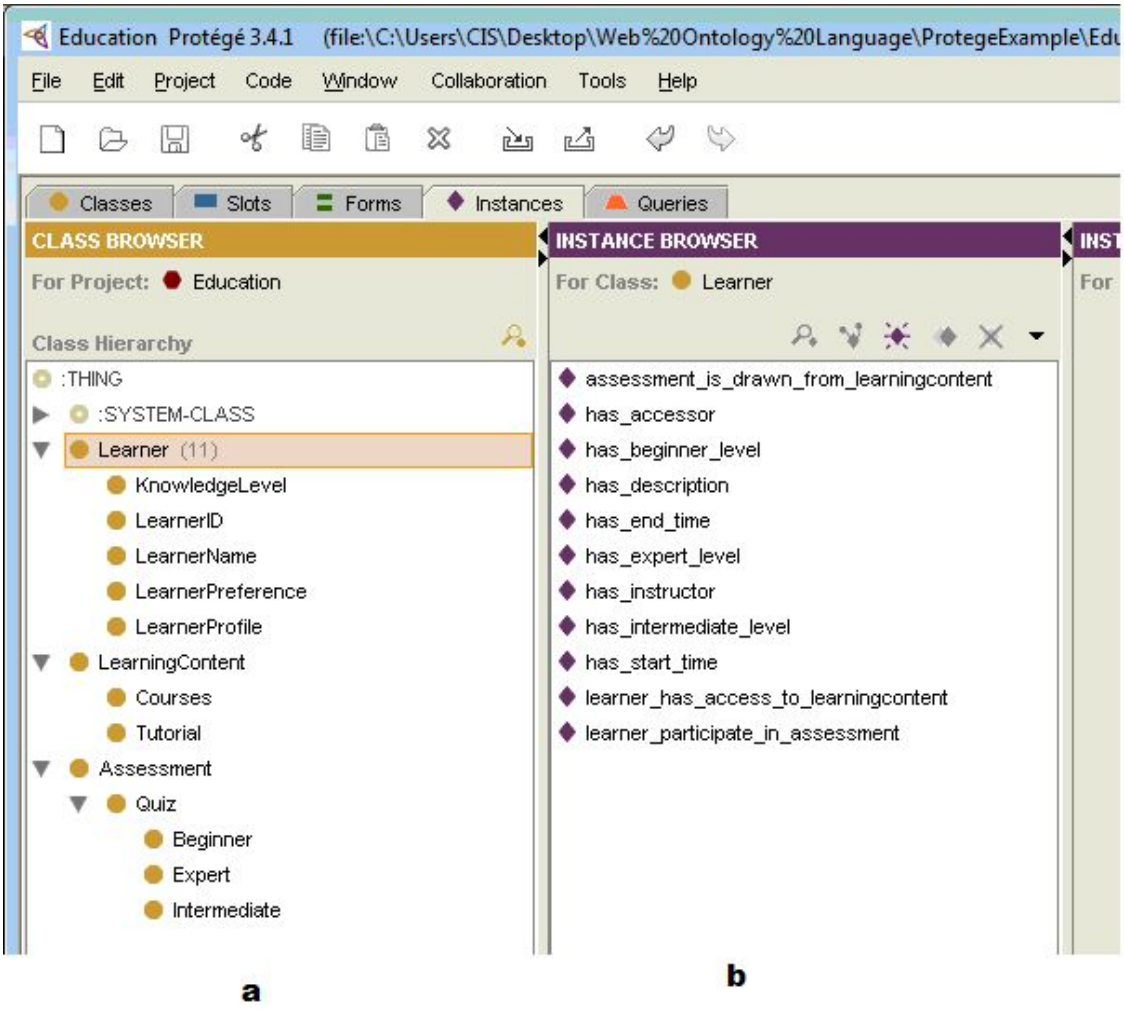
Entre las funciones más importantes, se encuentra el apartado de Classes, en donde se podrá situar los conceptos que definirán la ontología, la interfaz se muestra en la Imagen 2. Vista Classes, donde se pueden apreciar las opciones de agregar subclase, equivalencias, miembros, instancias, entre otros. Del dashboard de Protégé

Imagen No.2 Vista Classes



A partir de las opciones que nos da Protégé para definir las clases y subclases se puede llegar a realizar la estructura de una ontología sobre un tema específico, lo único que faltaría para completar la ontología es la agregación de instancias que representen la clase creada, para ello existe el campo Instances en donde se podrá agregar. En la Imagen 3 Screenshot of education ontology design using Protégé 3.4.1 se puede observar una ontología de educación, con su respectiva estructura e instancias.

Imagen 3. *Screenshot of education ontology design using Protégé 3.4.1*



(Azeta, Ayo and Omoregbe 2019)

1.5.2.8 Elasticsearch

Elasticsearch es un motor de búsqueda y análisis RESTful distribuido capaz de abordar volúmenes de datos muy grandes. Es una base de datos No Relacional. Su uso es por medio de una API, es orientada a documentos, utiliza JSON, no utiliza esquemas y permite búsquedas estructuradas y no estructuradas. (elastic 2019)

Tiene una gran variedad de clientes en diferentes lenguajes, como Java, php, python, Go entre otros. Sus consultas utilizan Query DSL basado en JSON para definir los *querys*, este software implementa consultas para identificar un valor

particular en un campo en particular, además de *querys* compuestos que permiten uso de una lógica más compleja con el objetivo de tener resultados más precisos. Cabe aclarar que estos *querys* tiene un comportamiento diferente dependiendo del contexto de la búsqueda y el contexto del filtro.

Al ser un motor que utiliza una API sus consultas son bastante legibles y concisas, en la *Imagen 4. Query elasticsearch Rest API* se muestra el ejemplo de un query básico de la búsqueda de un usuario en una base de datos

Imagen 4. *Query elasticsearch Rest API*

```
GET localhost:9200/_search?q=john
```

(Hasan 2019)

Al realizar la consulta anterior se muestra en la *Imagen 5. Respuesta Query elastic* la estructura de una respuesta común en elasticsearch de los documentos que incluyan “john”

Imagen 5. Respuesta Query elastic

```
{
    "took": 58,
    "timed_out": false,
    "_shards": {
        "total": 5,
        "successful": 5,
        "failed": 0
    },
    "hits": {
        "total": 2,
        "max_score": 0.2876821,
        "hits": [
            {
                "_index": "accounts",
                "_type": "person",
                "_id": "2",
                "_score": 0.2876821,
                "_source": {
                    "name": "John",
                    "lastname": "Smith",
                    "job_description": "Systems administrator"
                }
            },
            {
                "_index": "accounts",
                "_type": "person",
                "_id": "1",
                "_score": 0.28582606,
                "_source": {
                    "name": "John",
                    "lastname": "Doe",
                    "job_description": "Systems administrator and Linux
specialist"
                }
            }
        ]
    }
}
```

(Hasan 2019)

En este caso el resultado son 2 documentos que contenían el nombre “john”, además se puede apreciar los datos de la búsqueda como el tiempo que llevó, los campos que revisó y su respectivo estado, además se puede identificar la estructura de la respuesta de las búsquedas.

En contraste, se realizó un *query* utilizando lógica y una sintaxis más compleja, como se muestra en la *Imagen 6. Query Using DSL*, con el cual se quiere buscar toda la información referente a una obra de teatro, luego se debe contar y reunir todos los tipos de registros que existan sobre esa obra, además se debe agrupar

esta información en el campo “Total plays” y los datos de la puesta en escena de la obra se debe agrupar en el campo “Per type”.

Imagen 6. *Query Using DSL*

```
GET localhost:9200/shakespeare/_search
{
    "size":0,
    "aggs" : {
        "Total plays" : {
            "terms" : {
                "field" : "play_name.keyword"
            },
            "aggs" : {
                "Per type" : {
                    "terms" : {
                        "field" : "_type"
                    }
                }
            }
        }
    }
}
```

(Hasan 2019)

Luego de realizar el *query* se obtiene parte el resultado mostrado en la *Imagen 7. Response of DSL query*, en donde se puede apreciar que en el campo “Total plays” está agrupada toda la información respecto a la obra “Hamlet”, que a su vez tiene un campo de agrupación “Per type” el cual contiene la suma de todos los datos de la puesta en escena, como son: “escenas”, “líneas”, actores, entre otros.

Imagen 7. Response of DSL query

```
"aggregations": {
    "Total plays": {
        "doc_count_error_upper_bound": 2763,
        "sum_other_doc_count": 73249,
        "buckets": [
            {
                "key": "Hamlet",
                "doc_count": 4244,
                "Per type": {
                    "doc_count_error_upper_bound": 0,
                    "sum_other_doc_count": 0,
                    "buckets": [
                        {
                            "key": "line",
                            "doc_count": 4219
                        },
                        {
                            "key": "scene",
                            "doc_count": 20
                        },
                        {
                            "key": "act",
                            "doc_count": 5
                        }
                    ]
                }
            },
            ...
```


(Hasan 2019)

1.6 ESTADO DEL ARTE

En la exploración y búsqueda de estudios, investigaciones, proyectos y artículos similares, a través de base de datos para encontrar algunos de los avances más importantes respecto a este tema, se encontraron los siguientes proyectos:

Medibot: Un chatbot basado en ontología para personas que usan medicamentos y hablan portugués, recopila información acerca de los diferentes medicamentos sus costos y posibles riesgos para responder a las diferentes preguntas en lenguaje natural en portugués a través del servicio de mensajería instantánea: Telegram.Utiliza OWL y Link data para estandarizar diferentes DataSets con múltiples estructuras y vocabularios y ofrecer una capa de abstracción que permite acceso a los datos de una manera más fácil con el vocabulario que utiliza un usuario en su vida diaria. En este caso hacen uso de un booklet como fuente de información llamado “What we should know about drugs”. (Calixto and Magalhães 2019)

También existen diferentes enfoques de los chatbots a otras áreas como el e-commerce, donde se creó un chatbot que manejan todas las peticiones de los usuarios interesados en comprar dentro de la aplicación. (Vegesna, Jain and Porwal 2019). La plantilla de ontología fue desarrollada con Protégé, que almacena los conocimientos adquiridos de las API del sitio web, mientras que el administrador de diálogo se maneja con Wit.ai y la integración de ambos se realiza mediante Python. Utilizando una ontología enfocada a identificar las entidades y las propiedades que en este caso son todos los productos del e-commerce, logra superar algunas de las principales fallas de los chatbots y al mismo tiempo ofrece a los usuarios del ecommerce una mejor experiencia de usuario. (Vegesna, Jain and Porwal 2019).

En la educación también se han creado proyectos de este tiempo como este Chatbot(Lombardi, Colace and Pascale 2019) que ofrece un sistema de apoyo a la educación para los estudiantes. El propósito de este proyecto fue gestionar la comunicación y proporcionar las respuestas correctas al estudiante. Utilizan una ontología que es capaz de abordar la relación y evaluar la distancia semántica entre los objetos de aprendizaje, y de inferir la intención de un usuario y desambiguar la consulta (Lombardi, Colace and Pascale 2019).

El PLN es un sub-campo de la inteligencia artificial que le permite a las computadoras entender y usar el lenguaje natural humano. Muchas aplicaciones, como la detección de spam por correo electrónico, máquinas contestadoras de preguntas y traducción automática, chatbots, extracción de información y más (Khatter et al. 2019), utilizan PLN para aumentar su eficacia. Algunos de los proyectos que han utilizado estas tecnologías combinadas es un Control por voz para una Smart Home utilizando IoT (Jasmin Rani et al. 2019). Un usuario puede dar órdenes a la aplicación a través de un discurso de estilo libre, que luego son ejecutados por la aplicación. Para entrenar el sistema utilizaron herramientas open source y APIs de terceros como Dialog Flow, permitiéndoles un prototipado rápido para reducir ciclos de desarrollo y tener una versión más rápida en el mercado.

Una de las herramientas más utilizadas y reconocidas por su combinación de inteligencia artificial y poder computacional para realizar procesamiento y comprensión del lenguaje natural es IBM Watson Asistant, una solución analítica de extremo a extremo para ayudarnos a obtener información sobre nuestros datos, fue diseñada para que los científicos de datos, los desarrolladores de aplicaciones y los expertos en la materia trabajen de manera colaborativa y sencilla con los datos para construir y entrenar modelos a escala.(Hayes 2019).

Por otra parte, actualmente en Colombia, existen iniciativas que utilizan la tecnología para ayudar al campo. Entre ellas, existe una llamada MilAgro, la cual

tiene como objetivo la educación y capacitación de los campesinos en el uso de las tecnologías de la información y comunicación a través de Escuelas campesinas Digitales, Este modelo educativo está integrado por tres componentes principales:

- Primero, se capacita a campesinos de todas las edades poniendo a su disposición una plataforma virtual compuesta por cinco módulos de formación: alfabetización digital, conocimiento del medio, liderazgo, asociación y empresa, construcción de paz y convivencia.
- Segundo, se gestiona y alimenta una página web donde convergen todos los contenidos y oportunidades de desarrollo rural.
- Finalmente, a través del periódico “El Campesino”, se estimula a los campesinos a participar en la redacción de artículos y análisis de temas de interés agrícola.(Perdomo B. 2019)

1.7 METODOLOGÍA

Para la metodología se optó por basarse en SCRUM, adoptando el esquema de sprints para el entrenamiento del software de NLU y para la creación del método. Adicionalmente, se decidió obviar el proceso de reportes y se llegó al acuerdo de que este Documento de trabajo de grado es el único reporte, también se omitió el proceso de reuniones diarias y los roles propuestos por SCRUM (*Product Owner, Scrum Master* y *Scrum Team*).

Imagen 8. Diagrama de Metodología

1.7.1 Fases del proyecto

1.7.1.1 Fase inicial

La fase inicial del proyecto consta de la exploración de las características de los softwares elegidos por la empresa (IBM watson, Google Dialog Flow y LUIS) que ofrezcan las mejores características en la tarea de comprensión del lenguaje natural y que pueda ser implementado en la infraestructura actualmente creada. El voicebot requiere entender el contexto y el significado de la palabra que habló la persona y en base a eso poder generar un análisis para determinar cuáles son las entidades e intenciones de la persona con su pregunta y en base a estos resultados poder generar un query de búsqueda en la base de datos que contiene el audiolibro para obtener una respuesta más precisa y correcta.

Para poder realizar el proceso de análisis y comprensión del lenguaje natural se busca implementar librerías o software que ya existan en el mercado y que ayude a realizar el proceso mucho más rápido y eficientemente. La herramienta que se vaya a elegir puede ser de software libre o comercial, pero si es comercial no puede exceder el presupuesto estipulado por la empresa.

1.7.1.2 Fase de elaboración

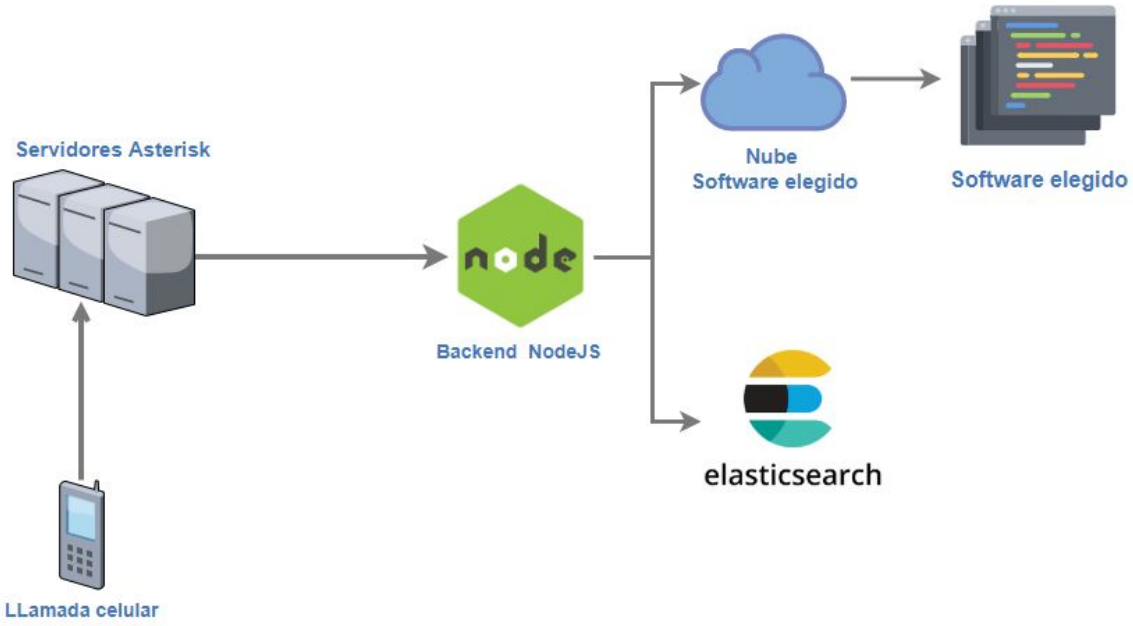
Se establecieron las funciones a realizar para la creación del proyecto en el horario de trabajo para cada uno de los integrantes del proyecto y se decidió adoptar la metodología interna de desarrollo de la empresa en este caso: SCRUM, donde se plantea aproximadamente 8 a 15 semanas de trabajo que cubren las siguientes actividades:

- Diseñar, implementar o presentar mejoras en el sistema de compresión y análisis del lenguaje natural en el *voicebot* utilizado actualmente por la empresa.
- Crear un modelo personalizado de datos u ontología en el área de dominio en este caso el cultivo de maíz, para ser utilizado por el software PLN seleccionado previamente.
- Entrenar el modelo de *machine learning* del software seleccionado para que identifique las entidades, relaciones, palabras claves y otras características del procesamiento del lenguaje natural en el área de dominio.
- Probar y testear las nuevas implementaciones al software verificando su efectividad y precisión.

1.7.1.3 Fase de verificación

En esta fase se busca que el software cumpla con las condiciones planteadas. Con base al análisis de herramientas realizado y con las herramientas ya seleccionadas, se verifica con la *Imagen 9. Diagrama del software* se cumpla:

Imagen 9. Diagrama del software



A continuación, se enumeran los pasos que debe realizar el software y deben ser cumplidos en su totalidad:

1) La persona va a llamar por celular o teléfono a los servidores de telefonía para conocer información sobre el audiolibro.
2) Los servidores Asterisk de telefonía reciben el audio captado por la persona y lo convierten a texto mediante la API de reconocimiento de voz para enviar el mensaje al servidor de nodejs.
3) El servidor en NodeJS recibe el mensaje de la persona en forma de texto y procede a realizar las consultas a la API para realizar el procesado y análisis de la frase u oración y poder obtener de vuelta el análisis.
4) El software de análisis de lenguaje natural identifica las entidades y relaciones de la oración, gracias al entrenamiento que tuvo el modelo de machine learning con el que cuenta la herramienta.

5) El servidor NodeJS con el análisis recibido, genera una consulta estructurada a la API de Elasticsearch para buscar en la base de datos la información.
6) El motor de Elasticsearch contiene toda la información indexada del audiolibro para realizar algoritmos de búsqueda y devolver la lista de posibles capítulos que tenga información acerca de los que busca la persona.
7) Con la lista de capítulos de vuelta, el servidor de NodeJS devuelve esta información al servidor de Asterisk para que reproduzca el audio a la persona con la respuesta a su consulta.

1.7.2 Instrumentos o herramientas utilizadas

Las herramientas que se van a utilizar en la construcción de este proyecto son:

- NodeJS
- PHP
- Asterisk
- IBM Watson Assistant
- ElasticSearch

1.8 DESARROLLO DE LA PROPUESTA

1.8.1 Presupuesto

Para el desarrollo del proyecto, la compañía tiene un presupuesto de $80 dólares para la suscripción mensual del software elegido. Teniendo en cuenta que se pretende que en promedio unos 2.000 usuarios al mes utilicen el servicio. A continuación, en la *tabla 2. Presupuesto general* se muestra el presupuesto desglosado por partes.

Tabla 2. Presupuesto general

.

Categoría	Recurso	Descripción	Cantidad	Costo(US)
Infraestructura	Servidor Asterisk	Es el servidor encargado de gestionar los datos referentes a la llamada (Geolocalización, Voz, número telefónico, flujo, entre otros.)	1	$0
	Servidor Backend	Es el servidor encargado de realizar las peticiones para realizar el análisis de la voz, la búsqueda necesaria y la respuesta. Las peticiones posibles son para: ● Software elegido (Análisis) ● Elasticsearch	1	$0

		(Búsqueda) ● Servidor PHP (Respuesta)		
	Servidor ElasticSearch	Es el servidor encargado de realizar la búsqueda en el audiolibro, teniendo en cuenta el análisis realizado en el software elegido	1	$0
Software	IBM Watson, Dialog Flow o Luis	Uno de los posibles candidatos para realizar el análisis del lenguaje, el cobro de los tres se realiza por número de peticiones. Se debe elegir únicamente un software	20000 por mes	$23 a $40
			Total Por mes (US)	$23 a $40

Como se evidencia, los elementos destinados a infraestructura, como el servidor de PHP, el servidor backend que gestionará todo el proceso y el servidor de Elasticsearch para la búsqueda, estarán cubiertos con los servidores actuales de la compañía y no tendrán ningún costo adicional reflejado en el proyecto, al igual que el audiolibro, el cual va a ser proporcionado a la empresa sin ningún coste adicional.

En conclusión, se estima que el proyecto tendrá un costo entre $23-40 dólares por mes teniendo en cuenta el software elegido y el promedio de 20.000 peticiones por usuario al mes, dividido en 2.000 usuarios al mes haciendo 10 repeticiones diarias en promedio.

1.9 PRODUCTOS A ENTREGAR

Al final el proyecto se entregará:

- Análisis comparativo entre los softwares candidatos para el reconocimiento de voz y para el análisis y procesado del lenguaje natural
- Ontología basada en el audiolibro y el lenguaje natural
- Método de clasificación en el software seleccionado
- Medidas del desempeño del método
- Documento del proyecto

2. ANÁLISIS COMPARATIVO ENTRE LAS HERRAMIENTAS COMERCIALES DE PLN

Los voicebots o asistentes virtuales de voz utilizan NLU para recibir oraciones dadas por una persona, estos reciben esta información para procesar la pregunta del usuario y generar una respuesta adecuada. Para que el bot pueda entender lo que dice la persona puede hacerlo mediante dos formas: la detección de patrones en las entradas del usuario o en la clasificación de intenciones mediante el procesamiento de lenguaje natural (PLN). La primera técnica es simple y fácil de implementar en la práctica, pero a gran escala la cantidad de entradas pueden variar demasiado, afectando su escalabilidad y funcionamiento. El PLN utiliza algoritmos de machine learning que clasifican cada entrada del usuario, utilizando un conjunto de datos como por ejemplo todas la posibles intenciones y entidades que pueden existir en cualquier entrada del usuario. Son difíciles de implementar a excepción que se haga mediante la ayuda de plataformas que implementen sus capacidades.

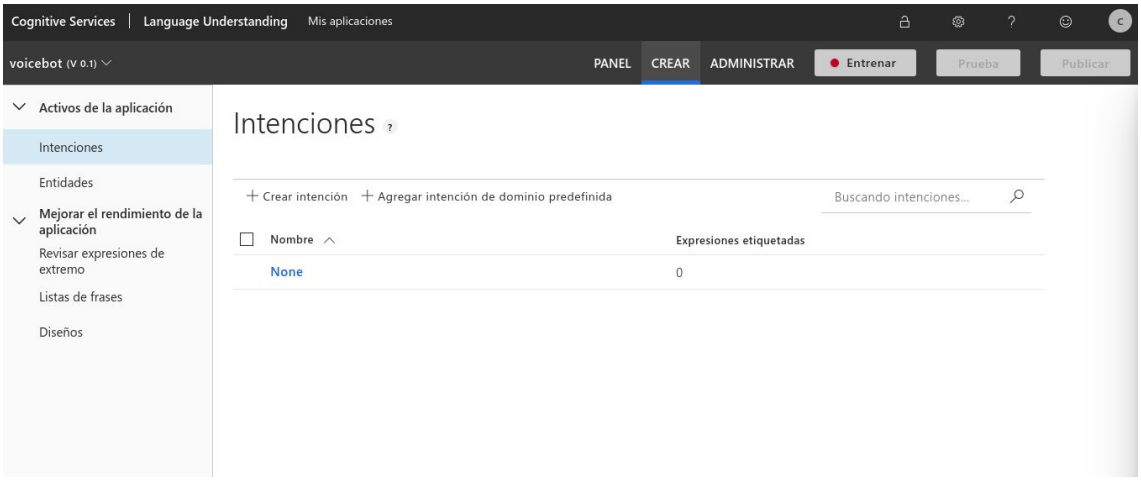
Algunas de las plataformas que existen en la actualidad son alojadas en la nube y exponen una interfaz gráfica para crear todo el modelo de predicción y de entrenamiento para los algoritmos de machine learning, también ofrecen comunicación de sus servicios mediante una API REST. Gigantes de la industria AI como IBM, Google y Microsoft saben que estas tecnologías son un campo muy importante para el futuro por eso cada uno cuenta con su propia plataforma:

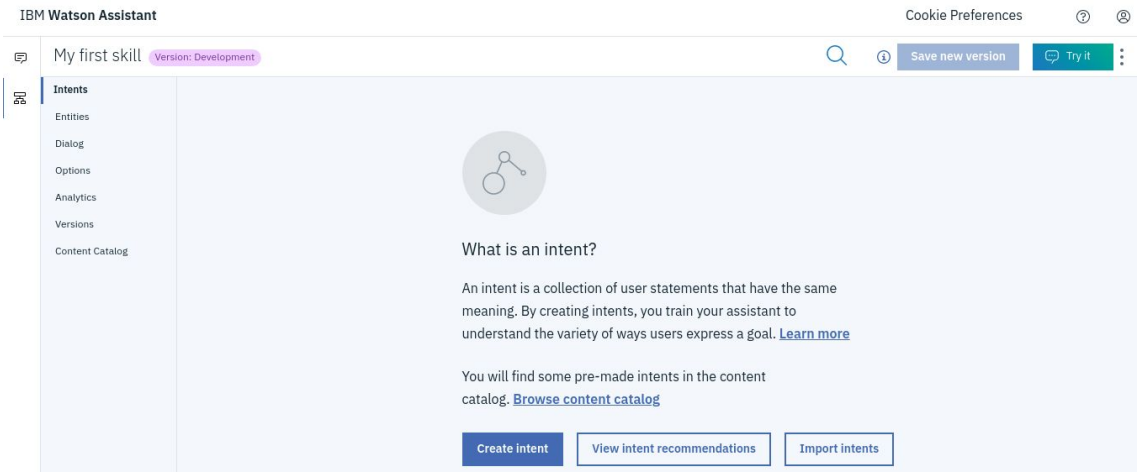
- Watson Assistant (IBM)
- LUIS (Microsoft)
- DialogFlow(Google)

2.1 Watson Assistant

IBM Watson Assistant es un servicio en la nube que permite crear un modelo de *machine learning* que clasifique intenciones por nosotros, y con este modelo crear chatbots cada vez más inteligentes y que aprendan automáticamente. Utiliza el aprendizaje automático de Watson AI (ML) y la comprensión del lenguaje natural (NLU).

Imagen 10. Entidades e intenciones en Watson





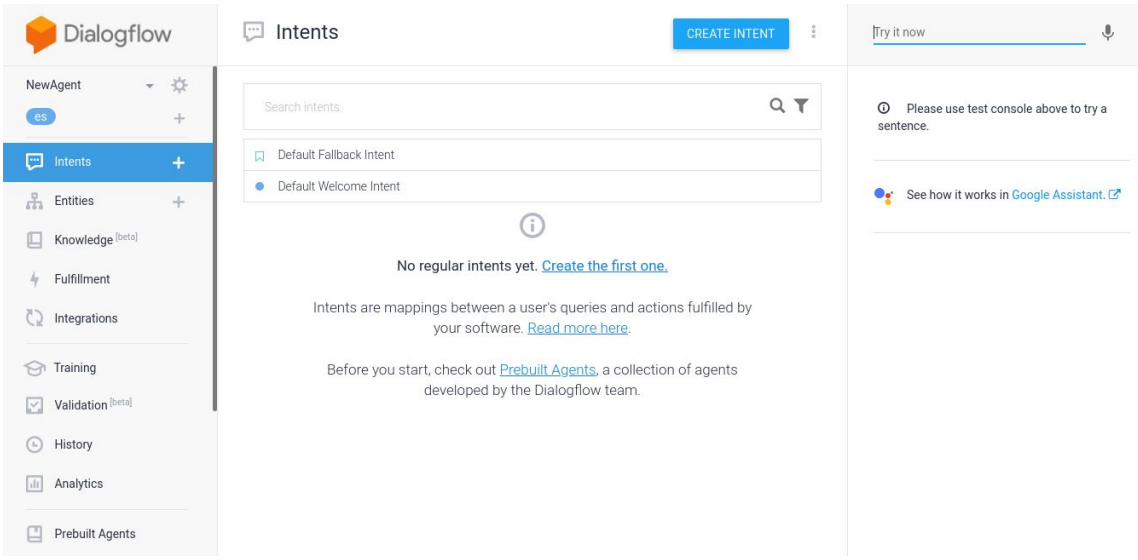
2.2 LUIS (*Language Understanding Intelligent Service)*

LUIS hace parte de los servicios cognitivos de Microsoft, basado en *Machine Learning* para crear una comprensión lingüística natural en aplicaciones, *bots* y dispositivos IoT. Crea rápidamente modelos personalizados preparados para la empresa que pueden mejorar constantemente.

2.3 Dialog Flow

Es la herramienta de Google para crear *chatbots* capaz de entender el lenguaje natural de los usuarios, con el objetivo de crear aplicaciones conversacionales en una gran cantidad de lenguajes a través de diferentes plataformas.

Imagen 11. Intenciones en DialogFlow



Estas plataformas simplifican bastante el proceso de creación de asistentes conversacionales, algunas proveen una solución integral mientras que otras tienen componentes que requieren integración mediantes sus API's. A continuación, se presenta una comparación entre las características más importantes de las tres plataformas.

2.4 MÓDULO DE ENTRENAMIENTO

Las tres plataformas ofrecen una interfaz gráfica para realizar todo el entrenamiento del modelo de *machine learning*, así como una interfaz donde se pueden probar el modelo ingresando ejemplos de entradas de un usuario y ver las clasificaciones de intenciones y detección de entidades. El entrenamiento consiste en ingresar todas las intenciones y entidades que fueron abstraídas de la ontología. Cada una de estas se define como:

- **Entidades**: Son combinaciones, sujetos, frases o palabras estándar que representan un significado sin instrucciones. Ej: “De qué color es el nuevo carro?”. La entidad en este caso es ***“carro”*** porque representa una entidad específica que tiene significado en el mundo real y puede ser utilizada en el dialecto humano. En la entrada de estas entidades también se puede agregar todos los sinónimos de palabras que pueda corresponder a esta entidad, así como un diccionario de palabras.

- **Intenciones**: Son las acciones generales que el usuario mediante su discurso quiso dar a entender. Ej: “quiero comprar un carro”. La intención será “comprar_carro”. Una vez la intención ha sido creada también se ingresa todo el conjunto de ejemplos en los que una persona pueda referirse a esta intención.

Aunque las tres permiten la entrada de entidades, solo Watson cuenta con intenciones pre construidas que automáticamente son detectadas si por alguna razón no fueron contempladas en el diseño inicial, así como pre intenciones de ejemplo o carga por defecto. Esto ayuda mucho ya que son intenciones muy bien entrenadas por el software que pueden ayudar a nuestro modelo de *machine learnin*g a clasificar mejor las entradas de los usuarios.

2.5 LENGUAJES

Dialogflow permite trabajar con 20 lenguajes, es el que software con más soporte lingüístico, Watson solo cuenta con soporte para 13 lenguajes y algunos aún están en la fase BETA, LUIS soporta 14 lenguajes, pero no todos tiene todas las características desarrolladas que necesita el software.

2.6 INTEGRACIONES

Las tres plataformas permiten la integración con herramientas sociales como Slack, Twitter y Facebook Messenger y también cuentan con la comunicación e implementación mediante el uso de sus correspondientes API’s, casa servicios dependiendo del plan de la cuenta registrada, limita la cantidad máxima de peticiones que son permitidas, esta se muestra más adelante en la tabla (nombre de la tabla). Sin embargo, Watson con su red de servicios cognitivos en la nube permite hacer la implementación de Watson Knowledge Studio, un servicio que

enseña a Watson Assistant el lenguaje de su dominio con modelos personalizados que identifican entidades y relaciones exclusivas de su industria en texto no estructurado.

2.7 CLASIFICACIÓN CORRECTA DE INTENCIONES

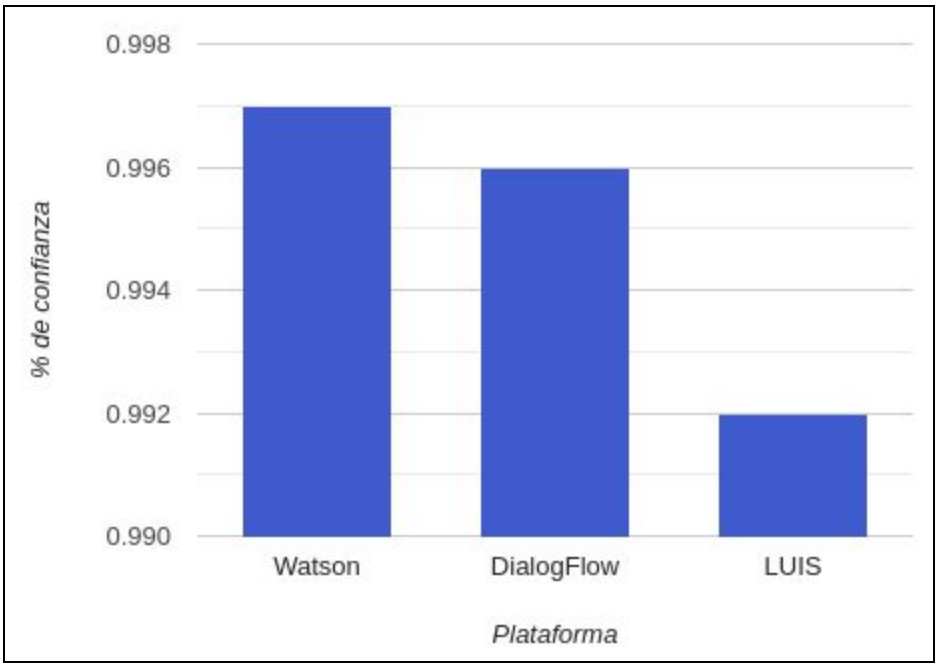
La parte más fundamental de estas aplicaciones es su clasificación correcta de intenciones, así el *bot* podrá entender correctamente qué es lo que la persona quiso decir y realizar la acción correcta. Para poder comparar qué aplicación tiene una mejor clasificación de entidades se necesita un dataset de entidades y por cada entidad proporcionar un set de frases de entrenamiento representando en una situación normal, el usuario como se referiría a esa intención. En 2017 una empresa de servicios cognitivos para aumentar la inteligencia en los negocios llamada Inten.to Inc, realizó una evaluación comparativa “NLU / Intent Detection Benchmark by Intento” de varias de estas plataformas incluyendo DialogFlow, LUIS y Watson. El *dataset* que se usó consistía de la siguiente información:

- Solo Ingles
- Siete intenciones cada una con 2000 frases de ejemplos para un total de alrededor de 15.600 ejemplos

Para cada plataforma se utilizó el dataset descrito y se entró el modelo para detectar todas las siete intenciones creadas. Algunos de los resultados que obtuvieron fueron los siguientes:

2.8 PRECISIÓN DE DETECCIÓN

Imagen 12. Gráfico de porcentaje de confianza



Se evidencia que los tres tiene un porcentaje de confianza demasiado alto para clasificar las intenciones correctamente, pero el ganador es Watson Assistant con 99.97% de confianza. Cabe aclarar que los que hicieron la prueba dicen que los resultados obviamente dependen de la data de las intenciones y que Watson Assistant es el mejor especialmente en un conjunto de datos pequeños. En conclusión, las tres plataformas tuvieron en general un muy buen puntaje de clasificación de las intenciones.

2.9 COSTO

Las tres plataformas ofrecen un plan gratuito de prueba y planes pagos, todas manejan precios similares del mercado, pero dependiendo de la escalabilidad que tenga el proyecto los precios pueden variar. Watson Assistant y LUIS permite realizar en el plan gratis 10.000 peticiones máximas al mes, mientras que Dialogflow permite realizar 180 peticiones por minuto. En sus planes pagos para

empresas Watson Assistant y DialogFlow cobra $0.002USD por cada petición mientras que LUIS cobra $1,50 USD cada 1000 peticiones.

A continuación, se presenta una tabla con todo el resumen de las características más importantes de las tres plataformas.

Tabla 3. Resumen de características generales de las Herramientas de PLN

	Dialogflow	IBM Watson	LUIS
Creador	Google	IBM	Microsoft
Módulo de Entrenamiento	✔	✔	✔
Entidades Preconstruidas	x	✔	x
Intenciones Preconstruidas	x	✔	✔
Número máximo de intenciones que se pueden crear	2000	2000	500
Número máximo de entidades que se pueden crear	250	1000	150
Mejor Clasificador de Intenciones	x	✔	x
Llamadas a la API	Planes: • Standard Edition:180	Planes: • Lite: 10.000 gratis al	Planes: • F0-Free tier: 10.000 gratis

	por minuto ● Essentials:600 por minuto ● Plus: 600 por minuto	mes ● Standard: ilimitadas pero vale $0.0025 USD cada petición ● Plus: precio preferencial por petición	al mes ● S0-Basic Tier: 50 por segundo ● Standard Tier: 50 por segundo
COSTO	Planes: ● Standard Edition: Gratis ● Essentials: $0.002 por petición ● Plus: $0.004 por petición	Planes: ● Lite: 10.000 gratis al mes ● Standard: $0.0025 USD por petición ● Plus: precio preferencial por petición	Planes: ● F0-Free tier: 10.000 gratis al mes ● Standard Tier: $1,50 USD por 1000 transacciones
LENGUAJES	20	13 pero 10 en BETA	14 pero solo 11 tiene la posibilidad de usarse para tener NLU

Según el análisis anterior, para el desarrollo de este proyecto se escogió utilizar Watson Assitant como herramienta de procesamiento de lenguaje natural por ser la herramienta que mejor clasifica las intenciones, tener módulo de entrenamiento, soportar el español y ofrecer un plan gratuito con las características suficientes para el desarrollo del proyecto.

3. DISEÑO DE ONTOLOGÍA

Para poder realizar correctamente el procesamiento del lenguaje natural sobre las preguntas de las personas, utilizando Watson Assistant, es necesario definir un vocabulario específico que limite el contexto del voicebot y a su vez pueda ser reutilizado entre las diferentes aplicaciones. Para definir este vocabulario se utilizó el concepto de ontología. "Una ontología define los términos a utilizar para describir y representar un área de conocimiento. Las ontologías son utilizadas por las personas, las bases de datos, y las aplicaciones que necesitan compartir un dominio de información (un dominio es simplemente un área de temática específica o un área de conocimiento, tales como medicina, fabricación de herramientas, bienes inmuebles, reparación automovilística, gestión financiera, entre otras) (Guzmán Luna, López Bonilla, & Torres, 2012).

Según el artículo "Metodologías y métodos para la construcción de ontologías" (Guzmán Luna, López Bonilla, & Torres, 2012). Existen diferentes métodos y metodologías para realizar el diseño e implementación de una ontología. Algunas de estas metodologías son:

Tabla 4. Metodologías para realizar el diseño e implementación de una ontología

Metodología	Descripción
CYC	Extracción de conocimiento de manera implícita para alimentar de manera cíclica la cada uno de los elementos que componen la ontología, usa herramientas de procesamiento de lenguaje natural o aprendizaje computacional.
USCHOLD Y KING	Identificar el propósito, establecer conceptos y términos, crear la ontología validar la ontología, reciclar la ontología en otros dominios. Se puede crear en el software *Enterprise Protect Artificial Intelligent application Institute* de la Universidad de Edimburgo en conjunto con IBM.
GRÜNINGER Y FOX	Identificar las aplicaciones posibles, seguido realizar preguntas en lenguaje natural para establecer el ámbito de la ontología estas preguntas logran extraer conceptos, propiedades, relaciones y axiomas.se utilizan las herramientas tales como: Enterprise Design Ontology, Project Ontology, Scheduling Ontology y Service Ontology.
KACTUS	construye la ontología sobre una base de conocimiento mediante la abstracción, mediante: a. especificar la aplicación, b. diseño previo con base a ontologías top level c. afianzamiento y estructuración de la ontología. Enterprise Toolset.

	Implementa estas ontologías mediante agentes usando herramientas “off-the-self” estilo plug-and-play.
METHONTOLOGY	Adapta la creación de ontologías con base a un proyecto informático, incluye fases como los son: la planificación, la calidad del resultado, la documentación, entre otros, está soportada en el entorno de desarrollo WebODE el cual contempla las etapas de especificación, conceptualización, formalización, implementación y mantenimiento.
ONTOLOGY DEVELOPMENT 101	Establecida por la Universidad de Stanford, está contempla a. Determinar el dominio y ámbito. b. Determinar la intención de uso c. Reutilizar ontologías o vocabularios existentes d. Enumerar los términos importantes del dominio. e. Definir jerarquía de clases. f. Crear las instancias.

(Guerrero Montaña 2019)

Para la creación del diseño de la ontología se decidido usar *ONTOLOGY DEVELOPMENT 101,* una metodología propuesta por la universidad de Stanford apropiada para aplicaciones de pequeña y mediana escala. Las ventajas de esta metodología son: reducir el tiempo y esfuerzo de desarrollo, reutilizando la misma ontología con su definición de vocabulario y dominio dentro de la aplicación de procesamiento y la base de datos. La metodología fue dividida entre tres fases: análisis, desarrollo e implementación.

3.1 FASE 1: ANÁLISIS

Para realizar el desarrollo de una ontología es recomendado siempre empezar analizando, cuales son las necesidades y causas por la cuales se va a desarrollar la ontología. Este proceso se hizo para definir su dominio y alcance, resolviendo las siguientes preguntas basadas completamente en el audiolibro.

- **¿Cuál es el dominio que va a abarcar la ontología?**

El dominio de la ontología de este trabajo está asociado al cultivo de maíz, tema en el cual es basado en el audiolibro: “*¡TU ASISTENTE A LA MANO! Manual del cultivo de maíz” por DEKALB*. La ontología sólo será basada con la información de este material.

- **¿Para qué se va a usar la ontología?**

El principal objetivo de la ontología es ayudar a crear una estructura de conceptos, que permita representar la información del audiolibro y que puedan ser compartida por las aplicaciones que se va utilizar el voicebot:

- **¿Para qué tipo de preguntas la información de la ontología va a brindar respuestas?**

La ontología debe responder a todas las posibles preguntas que el usuario pueda hacer relacionadas exclusivamente con el tema tratado en el audiolibro, como por ejemplo:

  - ¿Qué es una semilla?
  - ¿Cómo cultivar el maíz?
  - ¿Qué tratamiento puedo realizarle a la semilla de maíz?
  - ¿Cuáles son los tipos de semilla de maíz que existen?
  - ¿Qué plagas atacan al maíz?

- **¿Quién va a usar y mantener la ontología?**

La ontología será mantenida y actualizada por la empresa empleadora, además será utilizada por las siguientes aplicaciones:

➔ Elasticsearch como base de datos, utilizara la ontología para indexar la información según la estructura creada, de forma que sea más fácil responder con la parte del audiolibro correcta, la pregunta que el usuario hizo.

➔ La herramienta de procesamiento de lenguaje natural, utilizara la ontología para poder definir las entidades e intenciones en base a los conceptos y clases definidas en la ontología para agregar esa parte de contexto y alcance en el *voicebot*.

Después de realizar todo el análisis de requerimientos para la ontología, se definieron los términos generales más importantes en el audiolibro representados en la siguiente tabla.

Tabla 5. Términos generales más importantes en el audiolibro

No.	Término
1	maíz
2	semilla
3	sembrar
4	plagas
5	malezas
6	enfermedades
7	cultivo
8	agricultor
9	planta
10	agricultura

3.2 FASE 2: DESARROLLO

Para el desarrollo de la ontología se va a utilizar Protégé como ambiente de edición. Protégé es una plataforma gratuita de código abierto que proporciona un conjunto de herramientas para construir modelos de dominio y aplicaciones basadas en el conocimiento con ontologías (Stanford University 2019). Como primer paso para el desarrollo de la ontología utilizamos los términos o conceptos más claves e importantes dentro del audiolibro, que tienen un conjunto de elementos y propiedades en común para crear las clases y su jerarquía. La metodología establece tres formas de hacer lo anterior:

- *Top-down*: El desarrollo empieza con los conceptos más generales en el tema, luego con la especialización de dichos conceptos, por ejemplo si se especializa en vinos, existiría la clase vino y luego sus subclases, como vino tinto, vino blanco, vino rojo y otros.
- *Bottom-up*: El desarrollo empieza con las clases más específicas, luego estas clases se agrupan en conceptos más generales. Por ejemplo: Si se habla de vinos, se empezará definiendo las clases de “Pauillac” y “Margau”, posteriormente se crearía la subclase “bordeaux”.
- Combination: El desarrollo sería una combinación de conceptos *top-down* y *bottom-up*, debido a que se pueden definir conceptos principales y luego generalizar. También se puede empezar con algunos conceptos específicos y luego agregar conceptos más generales, incluso se podrían agregar conceptos intermedios para alguna clasificación específica. (F. Noy and L. McGuinness 2019)

En este caso se usó el enfoque top-down definiendo los conceptos más generales del tema, los cuales fueron maíz y agricultura como se observa en la figura.

Imagen 13. Conceptos usando top-down

Dentro del audiolibro gran parte de la información corresponde al maíz y a todas sus características para su cultivo, como las plagas que pueden existir y afectar un cultivo, las condiciones de crecimiento que las que debe contar, las enfermedades que puede adquirir, como es su proceso de recolección entre otros. Otra parte del audiolibro es dedicada a definiciones y temas generales sobre la agricultura moderna y la forma de impactar y beneficiar el cultivo de maíz.

3.3 FASE 3: IMPLEMENTACIÓN

Se exportó el diseño de la ontología a RDF desde Protégé y luego se convirtió a formato JSON para formar un archivo final con las clases de la ontología asociados a cada capítulo del audiolibro, y crear el modelo en la base de datos de ElasticSearch en el que la información es indexada correctamente.

Imagen 14. Gráfico de la ontología exportada de Protégé

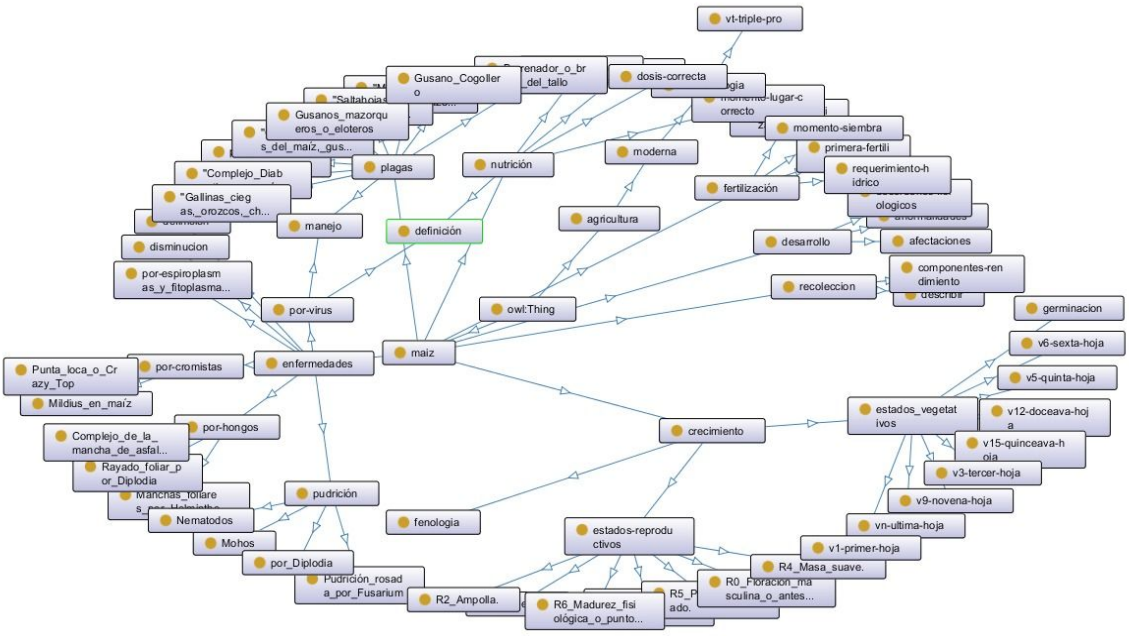
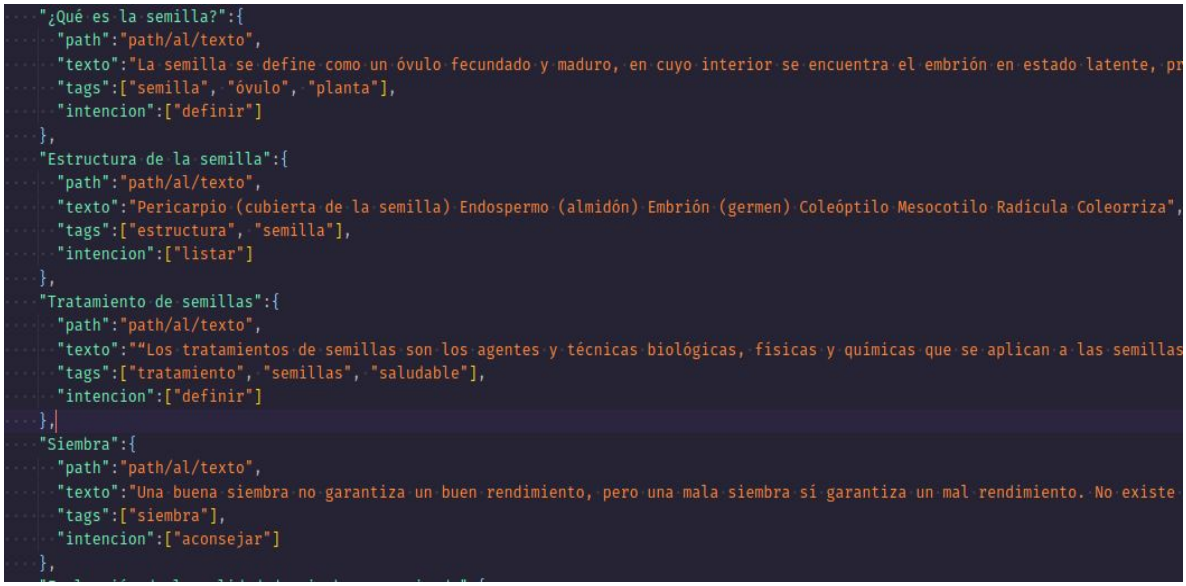


Imagen 15. Diseño de la ontología convertido a formato JSON



```
"¿Qué es la semilla?":{
  "path":"path/al/texto",
  "texto":"La semilla se define como un óvulo fecundado y maduro, en cuyo interior se encuentra el embrión en estado latente, pr
  "tags":["semilla", "óvulo", "planta"],
  "intencion":["definir"]
},
"Estructura de la semilla":{
  "path":"path/al/texto",
  "texto":"Pericarpio (cubierta de la semilla) Endospermo (almidón) Embrión (germen) Coleóptilo Mesocotilo Radícula Coleorriza",
  "tags":["estructura", "semilla"],
  "intencion":["listar"]
},
"Tratamiento de semillas":{
  "path":"path/al/texto",
  "texto":"“Los tratamientos de semillas son los agentes y técnicas biológicas, físicas y químicas que se aplican a las semillas
  "tags":["tratamiento", "semillas", "saludable"],
  "intencion":["definir"]
},
"Siembra":{
  "path":"path/al/texto",
  "texto":"Una buena siembra no garantiza un buen rendimiento, pero una mala siembra sí garantiza un mal rendimiento. No existe
  "tags":["siembra"],
  "intencion":["aconsejar"]
},
```

También se convirtió a un archivo CSV para poder importar todas las entidades dentro de Watson Assistant y poder reutilizar el diseño automáticamente. Así poder detectarlas en el procesamiento del lenguaje natural realizado en las preguntas de los usuarios.

4. MÉTODO

Para crear el método se usará una API desarrollada en NodeJS y Express. La API permitirá la integración entre aplicaciones y su intercambio de datos dentro del voicebot, de esta forma se puede asignar a cada aplicación una actividad específica dentro de la infraestructura. La lista completa de aplicaciones que conectara la API se presenta a continuación:

Tabla 6. Aplicaciones conectadas al API

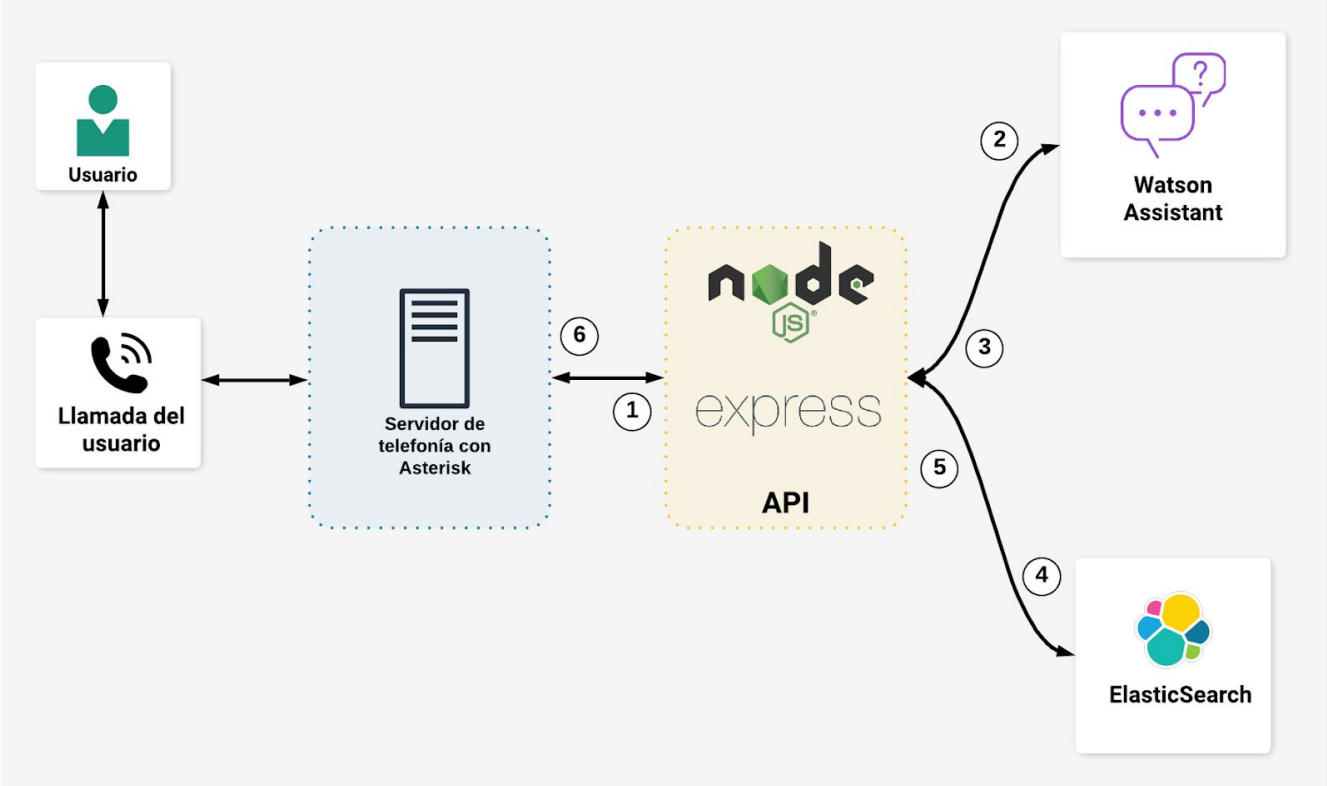
Aplicación	Funcion
Asterisk	Como plataforma de telefonía, se encargará de la gestión de llamadas entrantes y de las operaciones dentro de estás: captura de voz y números marcados.
IBM Watson Assistant	Con el entrenamiento terminado parcialmente, se encargará de clasificar las intenciones en las preguntas de los usuarios.
ElasticSearch	Base de datos donde se almacenará todas la información del audiolibro y único lugar donde se buscará las respuestas a las preguntas de los usuarios.

El diseño de la API por el momento es simple, para que su único objetivo sólo sea interconectividad entre aplicaciones y uso adecuado de todos los recursos. El funcionamiento completo del método propuesto con todas las aplicaciones es el siguiente:

1. Asterisk recibe la llamada entrante del usuario y en el momento programado, recibe la pregunta del usuario en lenguaje natural, que luego es convertida a una cadena de caracteres, para ser enviada a un endpoint específico de la API.

2. Cuando la API reciba la pregunta del usuario, enviará una petición a Watson Assistant para realizar el procesamiento de lenguaje natural y clasificar la intención de la pregunta del usuario, e identificar la entidad.
3. Watson Assistant retornara a la API el resultado compuesto por la intención de la pregunta con su porcentaje de confiabilidad, y la lista de entidades que existen dentro de la pregunta del usuario.
4. Con el resultado de la pregunta procesada, la API enviará una petición al endpoint de la ElasticSearch con el query de búsqueda para encontrar la información correspondiente.
5. Elasticsearch retornara como cadena de texto, el resultado que haya encontrado en su base de datos, con los parámetros que venían dentro de la consulta.
6. La API devolverá a Asterisk la respuesta de la base de datos, para que pueda ser reproducida en la llamada del usuario, con el fin de responder su pregunta formulada al inicio de la llamada.

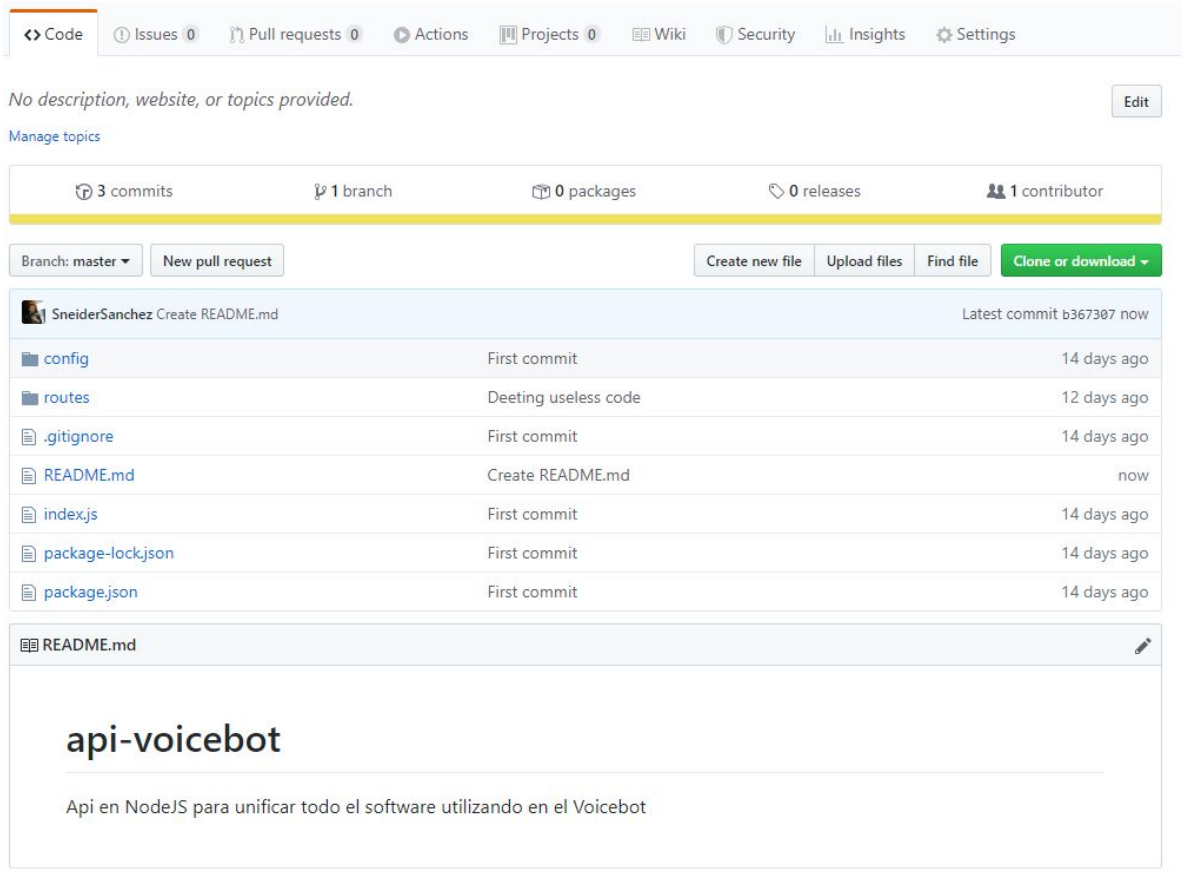
Imagen 16. Flujo de la información



4.1 CREACIÓN DE LA API

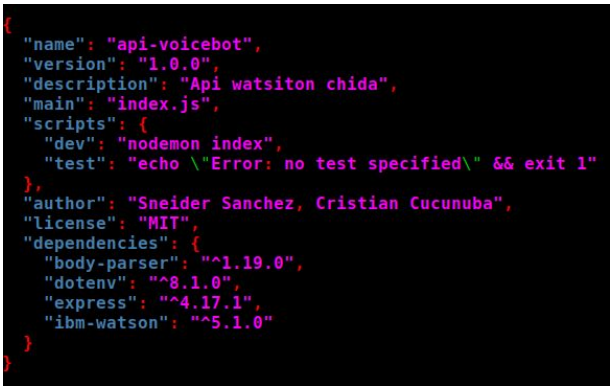
Para crear la API, primero crearemos un repositorio privado en Github como se muestra en la *Imagen 17. Repositorio en Github*. Esta herramienta se utilizó para manejar todo el versionamiento de código y colaboración del proyecto. Con el proyecto creado remotamente utilizamos Git para clonarlo en una máquina local y poder ser desarrollado desde ahí.

Imagen 17. Repositorio en Github.



Con el repositorio descargado en una máquina local, se continuó a crear el un nuevo proyecto de nodeJS con la ayuda de NPM mediante el siguiente comando en la terminal: npm -y api-voicebot. Para verificar que el proyecto se haya creado correctamente, se revisa que el archivo package.json, se haya creado automáticamente con los datos del proyecto correctos.

Imagen 18. Package.json de la API



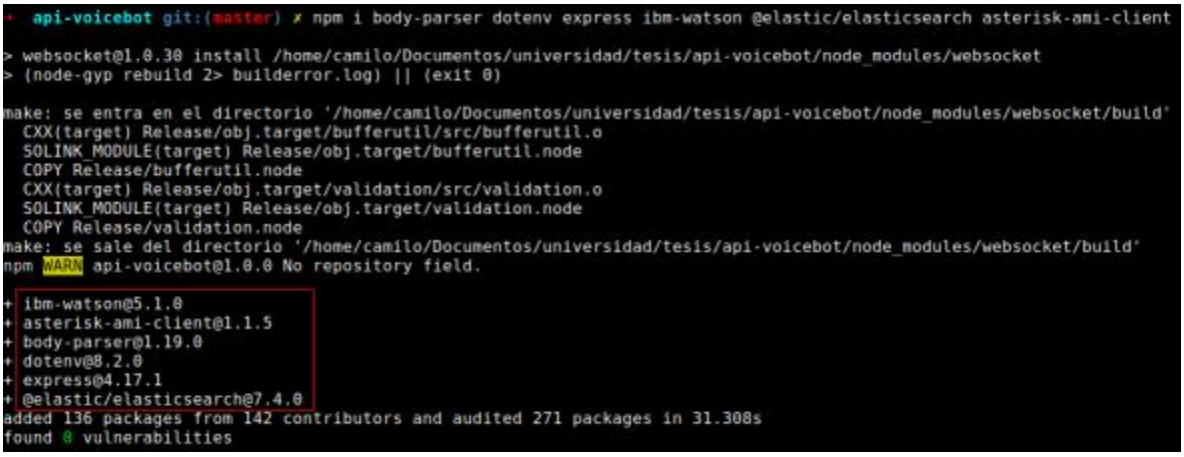
El IDE que se utilizó en este proyecto fue Visual Studio Code, por su gran soporte por la comunidad y experiencia de desarrollo tanto en el frontend como en el backend. Para el desarrollo de la aplicación también se necesitan librerías y paquetes externos de la comunidad, como los clientes de conexión con las demás aplicaciones y plugins con funcionalidades de JavaScript que son de utilidad. La lista de paquetes que se van a necesitar instalar con NPM en el proyecto son:

- **body-parser:** Permite parsear los datos que se envían a la API mediante peticiones de tipo POST.
- **dotenv:** Permite cargar las variables creadas en el ambiente por el archivo .env
- **express:** Web framework de Node para crear rutas, controladores, servicios entre otros.
- **ibm-watson:** Cliente para conectarse a Watson Assistant
- **@elastic/elasticsearch:** Cliente para conectarse a Elasticsearch
- **asterisk-ami-client:** Cliente para conectarse a Asterisk

Para instalar estos paquetes en el proyecto se utiliza el siguiente comando en la terminal:

> *npm i body-parser dotenv express ibm-watson* @elastic/elasticsearch *asterisk-ami-client*

Imagen 19. Instalación de paquetes del proyecto



Una vez finalice la instalación de estos paquetes como se muestra en la Imagen 20. Proyecto en Visual Studio Code e Imagen 21. Configuración del proyecto, se abre el proyecto con VS Code y se crea primero una carpeta config con un archivo index.js, el cual va a contener toda la información de configuración sensible de la API, como llaves de autenticación para las API públicas, puerto de arranque la API, URLs entre otros.

Imagen 20. Proyecto en Visual Studio Code

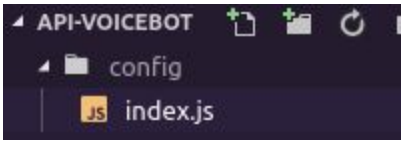
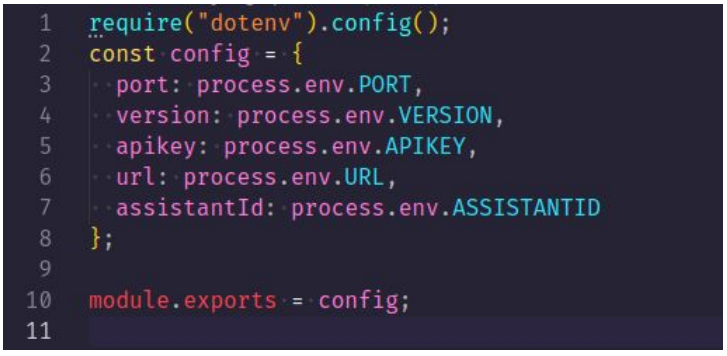


Imagen 21. Configuración del proyecto



```
require("dotenv").config();
const config = {
  port: process.env.PORT,
  version: process.env.VERSION,
  apikey: process.env.APIKEY,
  url: process.env.URL,
  assistantId: process.env.ASSISTANTID
};

module.exports = config;
```

Luego se crea una carpeta routes y dentro de esta carpeta, se crean tres archivos: asterisk.js, elasticsearch.js, watson.js. En cada archivo con ayuda de Express se declaran los endpoints o rutas que tendrá la API para comunicarse con las demás aplicaciones y la funcionalidad de cada una.

watson.js

Archivo con la ruta de tipo POST que recibe la pregunta y la envía a la API de Watson Assistant, como se observa en la línea 10 de la Imagen 22. Archivo watson.js, con el fin de recibir una respuesta en formato JSON con la intención de la pregunta y sus entidades como se muestra en la Imagen 23. Ejemplo de respuesta Watson.

Imagen 22. Archivo watson.js

```
1  const express = require("express");
2  const router = express.Router();
3
4  //Rutta para recibir la pregunta de la llamada y enviarla a
5  // Watson Asistant para que la procese
6  router.post("/process-question", async (req, res, next) => {
7    let watsonAnswer = "";
8    //conexion a watson assistant y envio de la pregunta a ser procesada
9    await req.app.locals.service
10       .message({
11         assistantId: "33022c0d-a730-4b33-b599-d1e65fabdf80",
12         sessionId: req.app.locals.session,
13         input: {
14           message_type: "text",
15           text: req.body.text
16         }
17       })
18       .then(res => {
19         //watson devuelve la pregunta procesada
20         watsonAnswer = JSON.parse(JSON.stringify(res.result));
21       })
22       .catch(err => {
23         console.log(err);
24       });
25   res.status(200).json({
26     message: "Text Analized ",
27     watsonAnswer: watsonAnswer,
28     error: false
29   });
30 });
31 module.exports = router;
```

Imagen 23. Ejemplo de respuesta Watson

Elasticsearch.js

El archivo con la ruta de tipo POST representado en la Imagen 23. Archivo elastic.js, que recibe la pregunta procesada en formato JSON, extrae la intención y las entidades para conectarse a la base de datos y hacer el query correspondiente. El resultado de la base de datos también es en formato JSON como se muestra en la Imagen 24. Ejemplo consulta en Elasticsearch y es enviado a asterisk para ser reproducido a la persona.

Imagen 24. Archivo elastic.js

```js
const express = require("express");
const router = express.Router();
//Conexion a Elastic Search
const { Client } = require('@elastic/elasticsearch')
const client = new Client({ node: 'http://localhost:9200' })

//Ruta que recibe la pregunta procesada y arma el query con la intencion
// y las entidades a la base de datos
router.post("/search:query", async (req, res, next) => {
  //Busqueda en la base de datos
  const { body } = await client.search({
    index: 'game-of-thrones',
    body: {
      query: {
        //Busca en el documento con la intencion correspondiente
        match: { intention: req.body.intention },
        //Busca en el documento que tenga tags parecidas a las entidades
        match: { entities: req.body.entities }
      }
    }
  })
  res.status(200).json({
    message: "Question Searched in the database",
    response: body.response,
    error: false
  });
});

module.exports = router;
```

Imagen 25. Ejemplo consulta en Elasticsearch


```
{
  "id" : "2",
  "title" : "¿Qué es la semilla?",
  "chapter" : "1",
  "tags" : [
    "semilla",
    "óvulo",
    "planta"
  ],
  "path" : "",
  "intencion" : [
    "definir"
  ],
  "message" : "La semilla se define como un óvulo fecundado y maduro, en cuyo interior se
    encuentra el embrión en estado latente, protegido por una cubierta llamada pericarpio y
    rodeado de reservas alimenticias conocidas como endospermo. La calidad de la semilla es el
    conjunto de características deseables referidas a la capacidad de la semilla para ser
    sembrada y dar origen a una nueva planta. Al evaluar la calidad de la semilla se
    consideran cuatro aspectos: genéticos, físicos, fisiológicos y fitosanitarios."
}
```

asterisk.js

Archivo con la ruta de tipo POST que recibe la respuesta de la base de datos, con la información que busca el usuario en formato JSON, para ser reproducida de vuelta en la llamada.

Imagen 26. Archivo asterisk.js

```
const express = require("express");
const router = express.Router();


/* Ruta que recibe la respuesta a la pregunta del usuario
y la reproduce de vuelta */
router.post("/question-user", (req, res, next) => {
  //Conexion con Asterisk
  const AmiClient = require('asterisk-ami-client');
  let client = new AmiClient({
    reconnect: true,
    keepAlive: true,
    emitEventsByTypes: true,
    emitResponsesById: true
  });
  /* Envio de la respuesta de la base de datos para que sea
  reproducida al usuario */
  client.connect('user', 'secret', { host: 'localhost', port: 5038 })
    .then(() => {
      client
        .on('Dial', event => console.log(event))
        .on('Bridge', event => console.log(event))
        .on('response', response => {
          //Reproducir respuesta de la DB
          client.play({ text: response.body })
        })
        .on('internalError', error => console.log(error));
    })
    .catch(error => console.log(error));
});

module.exports = router;
```

Para unir todas estas rutas y unificar el proyecto creamos el *entry-point* de la aplicación el archivo index.js. Este archivo arranca la aplicación y levanta el servidor HTTP, declara todos los *middlewares* que se van a usar, y llamas todas las rutas de la API.

Imagen 27. Archivo index.js

```
const express = require("express");
const bodyParser = require("body-parser");
const app = express();
const http = require("http").Server(app);
const { port, assistantId, version, apikey, url } = require("./config");
const watsonRouter = require("./routes/watson");
const elasticRouter = require("./routes/elasticsearch");
const asteriskouter = require("./routes/asterik");
const AssistantV2 = require("ibm-watson/assistant/v2");
const { IamAuthenticator } = require("ibm-watson/auth");
app.use(bodyParser.json());

//Levanta el servidor HTTP que escucha las peticiones a la API
const server = http.listen(port, () =>
  console.log(`Listening on http://localhost:${server.address().port}`)
);
//Middlewares para habilitar CORS
app.use((req, res, next) => {…
});
const service = new AssistantV2({…
});
// Rutas de la API
app.use("/api/watson", watsonRouter);
app.use("/api/elasticsearch", elasticRouter);
app.use("/api/asterisk", asteriskouter);
```

Con todos los pasos anteriores terminados, se corre la aplicación con el comando:

node index.js

Con esto la API ya está lista para escuchar peticiones HTTP e interconectar todas las aplicaciones con su información.

Indexación de la información del audiolibro en la base de datos

La base de datos del proyecto se va a utilizar elasticsearch. Es una base de datos NoSQL que permite guardar volúmenes grandes de documentos y ofrece uno de los mejores rendimientos de búsqueda en bases de datos en el mercado. Devuelve los resultados en formato JSON y se acopla integra fácilmente al stack desarrollado.

Como base de datos NoSQL en lugar de utilizar tablas y relaciones, hace uso de la siguiente estructura de información:

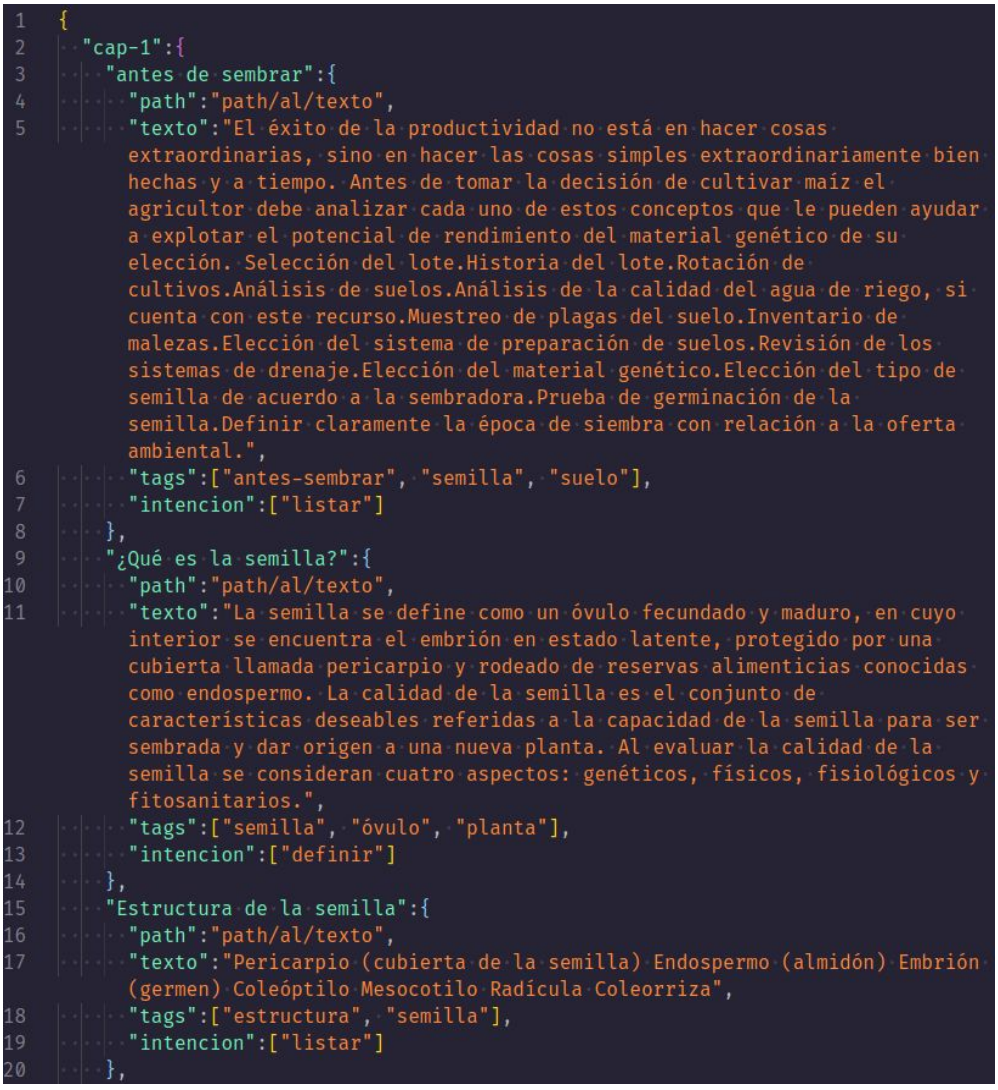
- **Index**: Una búsqueda en Elasticsearch nunca arroja el contenido como respuesta, sino el índice, en el cual se almacenan, ya preparados, todos los contenidos de todos los documentos. De esta forma la búsqueda requiere muy poco tiempo. Es el llamado inverted index (índice invertido): para cada término de búsqueda se indica el lugar donde se puede encontrar dicho término.
- **Document**: La salida para el índice son los documentos, en los cuales aparecen los datos. No tienen que ser necesariamente textos completos (por ejemplo, artículos de blogs) –es suficiente que se trate de archivos con información.
- **Field**: Un documento, a su vez, consta de varios campos. Además del campo de contenido propiamente dicho, hay otros metadatos que también forman parte de un documento. Por ejemplo, Elasticsearch puede utilizarse para buscar metadatos sobre el autor o el momento de la creación.(IONOS, 2019)

Para indexar la información se crea el *index "*audiobot" y dentro de este, el documento JSON con toda la información de cada capítulo:

- **nombre:** nombre del capítulo
- **path**: locación física del audio
- **texto**: texto completo del segmento del audiolibro
- **tags**: entidades representativas del capítulo y que se relacionan con las que identifica Watson Assistant en la pregunta de cada usuario
- **intención**: intención o acción que describe el segmento del audio y que se relaciona con la que clasifica Watson Assistant en la pregunta de cada usuario

En la siguiente *Imagen 27. Fragmento JSON audiolibro* se ve un fragmento del JSON con la información del capítulo uno del audiolibro

Imagen 28. Fragmento JSON audiolibro



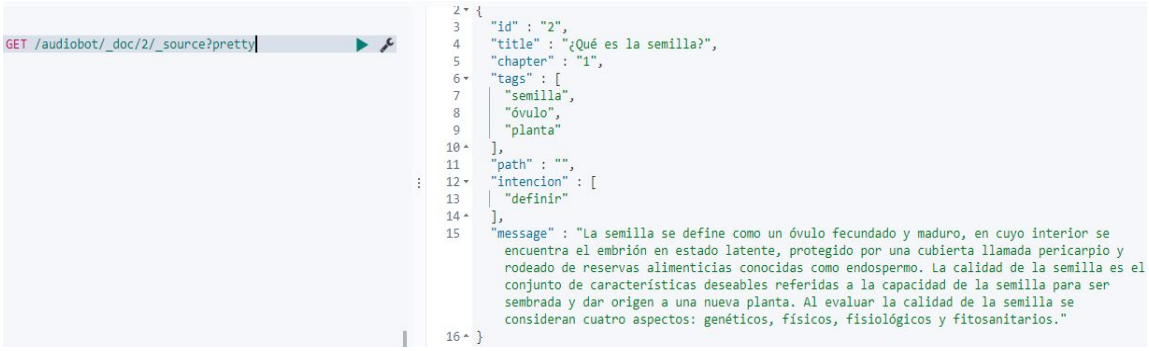
```
{
  "cap-1":{
    "antes de sembrar":{
      "path":"path/al/texto",
      "texto":"El éxito de la productividad no está en hacer cosas extraordinarias, sino en hacer las cosas simples extraordinariamente bien hechas y a tiempo. Antes de tomar la decisión de cultivar maíz el agricultor debe analizar cada uno de estos conceptos que le pueden ayudar a explotar el potencial de rendimiento del material genético de su elección. Selección del lote.Historia del lote.Rotación de cultivos.Análisis de suelos.Análisis de la calidad del agua de riego, si cuenta con este recurso.Muestreo de plagas del suelo.Inventario de malezas.Elección del sistema de preparación de suelos.Revisión de los sistemas de drenaje.Elección del material genético.Elección del tipo de semilla de acuerdo a la sembradora.Prueba de germinación de la semilla.Definir claramente la época de siembra con relación a la oferta ambiental.",
      "tags":["antes-sembrar", "semilla", "suelo"],
      "intencion":["listar"]
    },
    "¿Qué es la semilla?":{
      "path":"path/al/texto",
      "texto":"La semilla se define como un óvulo fecundado y maduro, en cuyo interior se encuentra el embrión en estado latente, protegido por una cubierta llamada pericarpio y rodeado de reservas alimenticias conocidas como endospermo. La calidad de la semilla es el conjunto de características deseables referidas a la capacidad de la semilla para ser sembrada y dar origen a una nueva planta. Al evaluar la calidad de la semilla se consideran cuatro aspectos: genéticos, físicos, fisiológicos y fitosanitarios.",
      "tags":["semilla", "óvulo", "planta"],
      "intencion":["definir"]
    },
    "Estructura de la semilla":{
      "path":"path/al/texto",
      "texto":"Pericarpio (cubierta de la semilla) Endospermo (almidón) Embrión (germen) Coleóptilo Mesocotilo Radícula Coleorriza",
      "tags":["estructura", "semilla"],
      "intencion":["listar"]
    },
```

Una vez el documento JSON esté completamente creado y terminado por cada capítulo, se utiliza la API de elasticsearch para subir el *index* con el documento a la base de datos con el siguiente comando en la terminal:

curl -X POST ‘localhost:9200/audiobot’-d ‘{nombre_archivo_json}’

Para verificar que el documento se haya creado correctamente en la base de datos, se realiza una búsqueda de ejemplo como se muestra en la *Imagen 28. Ejemplo búsqueda en Elasticsearch.*

Imagen 29. Ejemplo búsqueda en Elasticsearch



La base de datos ya está lista para hacer búsquedas con la información del audiolibro.

4.2 ENTRENAMIENTO DE WATSON ASSISTANT

Para entrenar a Watson Assistant se necesita subir la lista de entidades e intenciones. Las intenciones también tiene que llevar la lista de frases de ejemplo que representan la intención. Por ejemplo si la intención es *definir* una frase de ejemplo seria: "*Que es el maíz?".* Así el algoritmo clasificador se va a entrenando con el tipo de preguntas que puedan venir en un futuro.
Para subir la lista de entidades e intenciones primero se ingresa a la plataforma y se inicia la aplicación.

Imagen 30. Ingreso a Watson Assistant

Creamos el proyecto para iniciar una *skill,* que es donde se entrena el modelo y seleccionamos el lenguaje español para que reconozca el lenguaje de los ejemplos en cada intención, y realice su entrenamiento correctamente.

Imagen 31. Creación del *assistant*

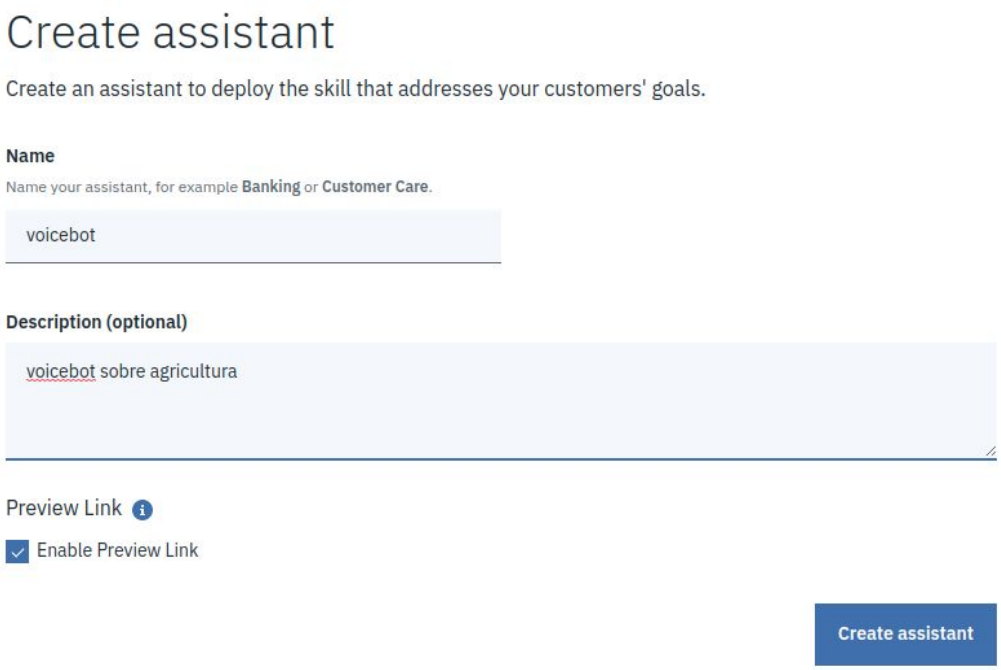
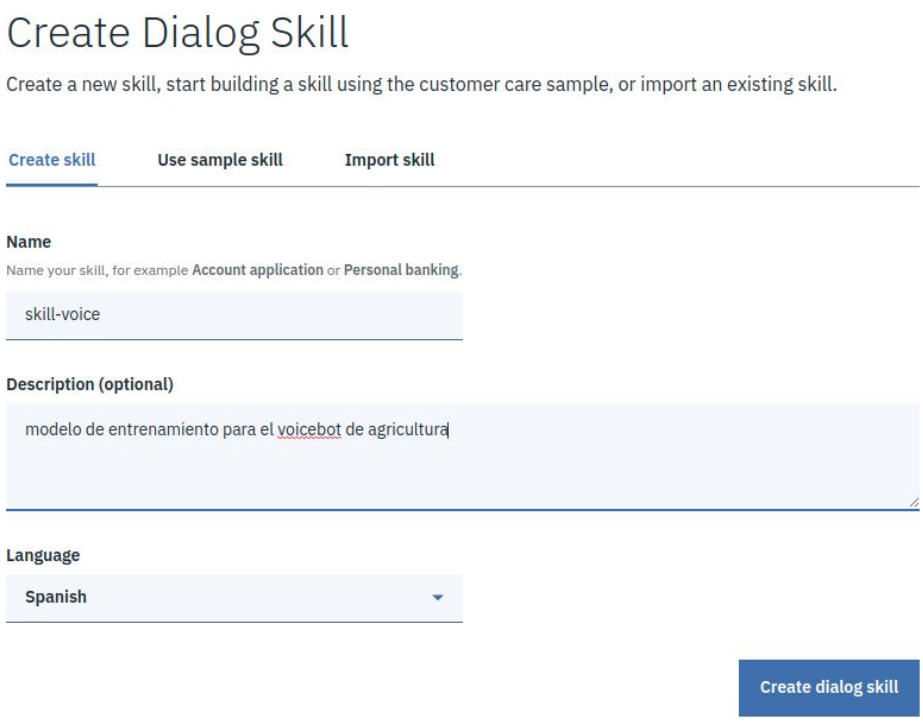


Imagen 32. Creación del *Skill*



Dentro del *skill* se importan los archivos en formato .csv que tienen las entidades y las intenciones con sus ejemplos para referirse estas, como se muestra en la *Imagen 32. Ejemplo de intenciones en formato .csv y en la Imagen 33. Ejemplo de entidades en formato .csv*. El proceso de importación es bastante sencillo y se muestra en la *Imagen 34. Ejemplo de importación de intenciones en Watson y en la Imagen 35. Ejemplo de importación de entidades en Watson.*

Imagen 33. Ejemplo de intenciones en formato .csv

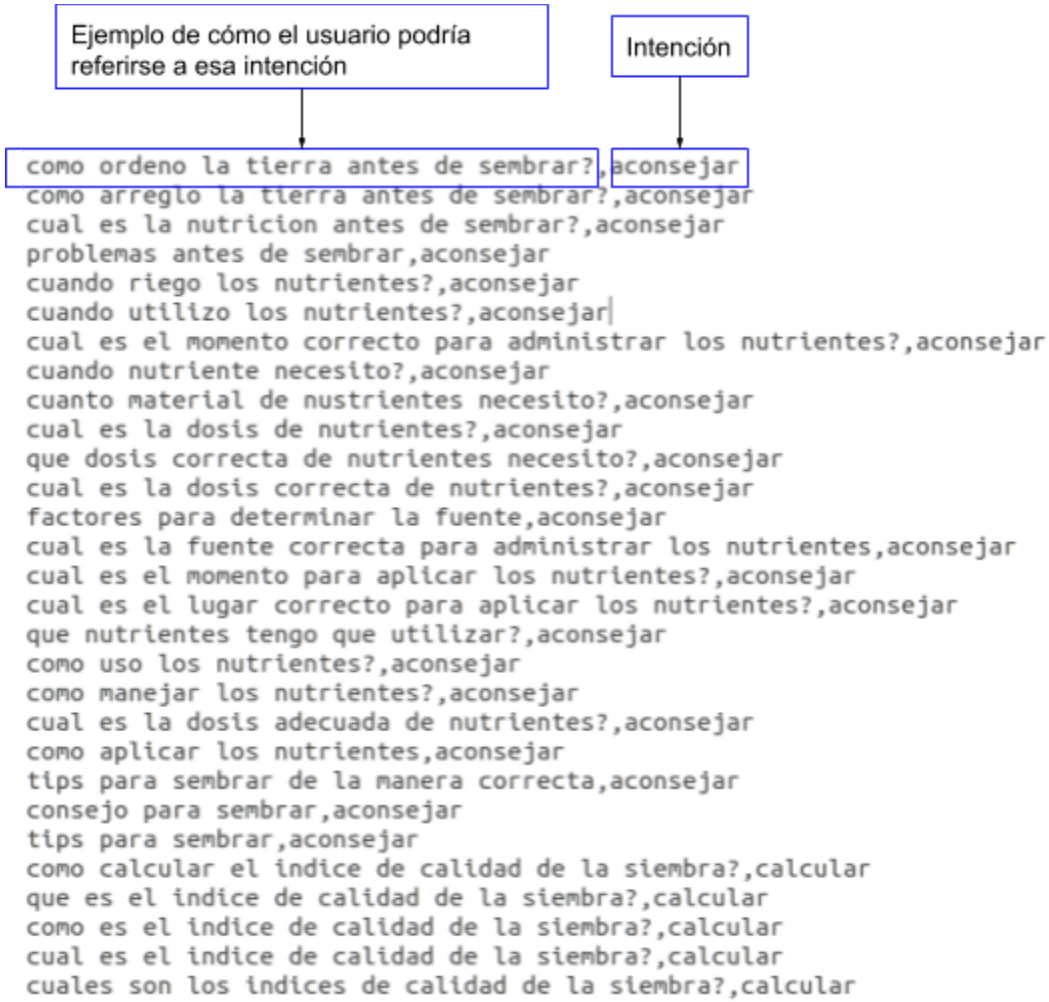


Imagen 34. Ejemplo de entidades en formato .csv

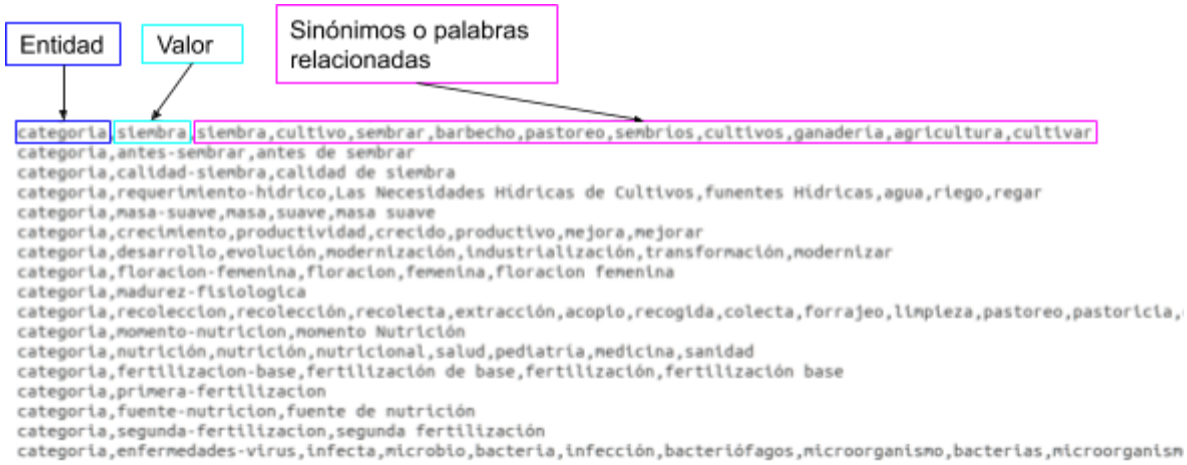


Imagen 35. Ejemplo de importación de intenciones en Watson

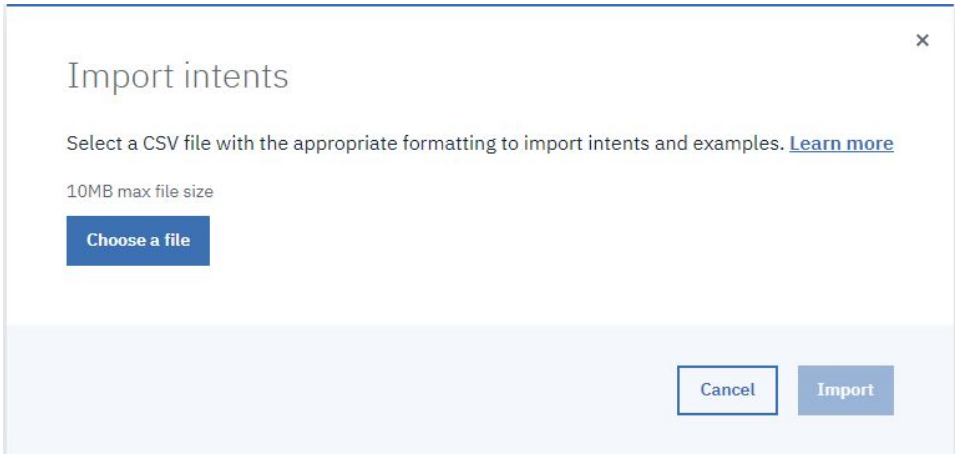
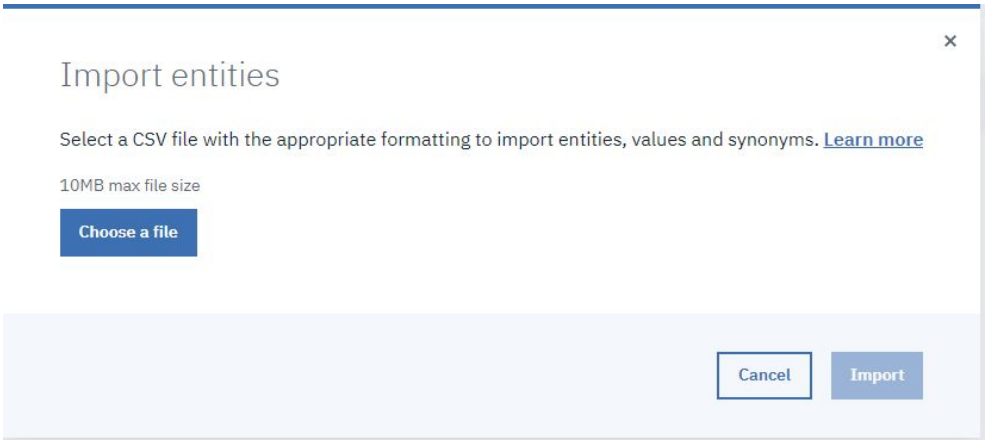


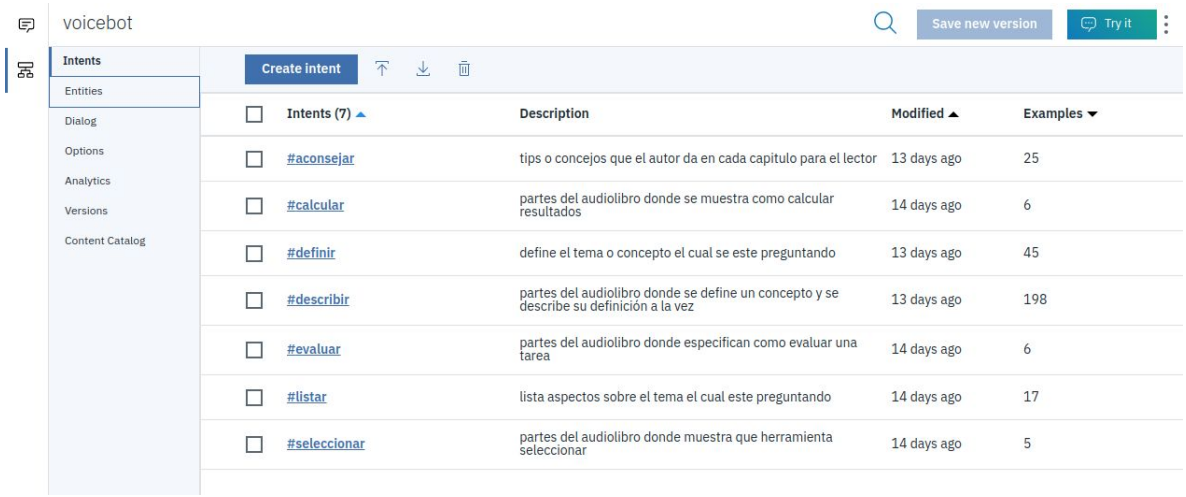
Imagen 36. Ejemplo de importación de entidades en Watson



Cuando se hayan importado correctamente los archivos, las intenciones y entidades van a estar dentro de la plataforma como se muestra en las siguientes imágenes:

Imagen 37. Vista de las entidades en Watson

Imagen 38. Vista de las intenciones en Watson



El modelo de machine learning es entrenado y actualizado automáticamente por Watson cuando detecta nuevos cambios. En este caso para probar que este funcionando correctamente, se hace uso de la opción "*try it*".

Imagen 39. Uso del módulo “*try it*”

Con la Watson assistant ya entrando, se puede hacer uno de su API para generar estos tipo de análisis durante el proceso de la llamada.

Integración

Con todas las herramientas listas y preparadas, se conecta el servidor Asterisk con la API para que le envíe la información necesaria, y está conecte automáticamente todas las aplicaciones de la forma explicada previamente, con el fin de responder la pregunta del usuario.

5. RESULTADOS Y ANÁLISIS DE RESULTADOS

Como se determinó en el análisis de herramientas en el capítulo 2, IBM Watson Asistant fue la herramienta escogida para realizar el procesamiento de lenguaje natural, por ser la herramienta de *machine learning* que mejor clasifica las intenciones para *datasets* pequeños en el mercado actualmente, así como por sus características como el módulo de entrenamiento y el módulo de pruebas gráfico. También durante el desarrollo del proyecto la empresa empleadora aprobó el uso de este software y estuvo de acuerdo en los planes y tarifas que puede tener a largo plazo.

La herramienta fue entrenada como se observa en el capítulo 4, con la ayuda del diseño de la ontología creada en el capítulo 3. Al final del entrenamiento Watson Assistant cuenta con 7 intenciones: definir, describir, evaluar, listar, seleccionar, aconsejar y calcular para un total de 300 ejemplos de preguntas que describen cada intención. También cuenta con dos entidades: “categoría” con 49 valores en los que se encuentran temas del audiolibro como: desarrollo, biotecnología, germinación, malezas entre otros, y “sujeto” con valores sobre sujetos, individuos, personas el cual cuenta 55 valores como semilla, maíz, hoja, gusanos entre otros.

Para validar el método construido en el capítulo anterior, se va a evaluar el modelo de clasificación de intenciones y reconocimiento de entidades de Watson Assistant con el entrenamiento recibido. Es necesario tener un *dataset* de pruebas con preguntas clasificadas por sus intenciones, las preguntas son sobre un tema y dominio muy específico y deben poder ser respondidas con la información del audiolibro.

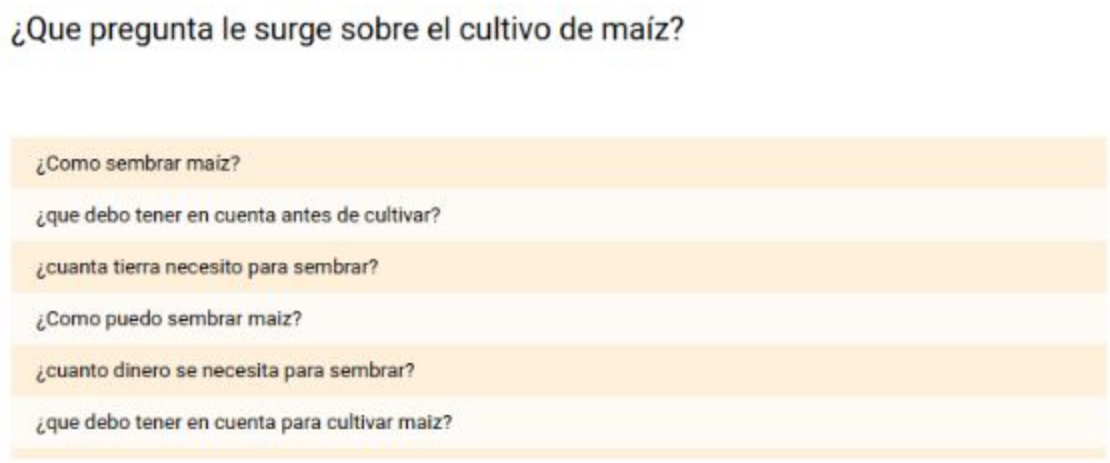
5.1 DATASET DE PRUEBA CLASIFICADO

Se buscó en internet un *dataset* que cumpliera estas características, pero no fue posible encontrar alguno. Es por eso que se optó por construir un experimento de pruebas, donde mediante el uso de un cuestionario, en donde cada pregunta está compuesta de una imagen alusiva a un tema del audiolibro, seguido de un espacio para que el usuario escriba una pregunta que le surja respecto al tema propuesto. se construirá un *dataset* de prueba propio.

El cuestionario fue desarrollado en la herramienta de Google Forms, está conformado por 15 preguntas y cada imagen de las preguntas fue extraída exclusivamente del libro original, en la Imagen 39. Fragmento del cuestionario en Google Forms se encuentra un ejemplo de las preguntas realizadas. Adicionalmente en la Imagen 40. Fragmento de las respuestas del formulario se encuentra un ejemplo de las respuestas obtenidas. El cuestionario fue enviado a otros estudiantes de la universidad por correo electrónico y se reunieron alrededor de 150 preguntas para construir el dataset y probar el método.

Imagen 40. Fragmento del cuestionario en Google Forms

Imagen 41. Fragmento de las respuestas del formulario



Se reunieron todas las respuestas de las personas, y se clasificaron por juicio de experto manualmente cada una. Para obtener al final el dataset completo clasificado por intenciones. Luego se pasó cada pregunta por la API y se obtuvo la respuesta, con el fin de verificar cuales son las intenciones y entidades que se obtienen y a partir de estos datos, se optó por crear una tabla por cada una de las preguntas.

La tabla está distribuida en 3 columnas de la siguiente manera: respuesta esperada, respuesta obtenida en Watson(formateada) y respuesta obtenida en Watson (completa), en la Imagen 41. Ejemplo tabla de respuestas para la matriz de confusión, se muestra un ejemplo de la tabla. En el Anexo 1 de este proyecto, se encuentran todas las tablas para las 15 preguntas.

Imagen 42. Ejemplo tabla de respuestas para la matriz de confusión

Tema: Recurso Hidrico	Pregunta:¿Que pregunta le surge sobre el recurso hídrico del maíz?		
	Respuesta esperada	Respuesta Watson (formateada)	Respuesta Watson (sin formatear)
¿Cuanta agua necesita mi cultivo?	**Entidad:** @categoria:requerimiento-hidrico **Intencion:** #describir	**Entidad:** @categoria:requerimiento-hidrico @categoria:siembra **Intencion:** #describir	intents: [{ "intent": "describir", "confidence": 0.9889055252075196 }], "entities": [{ "entity": "categoria", "location": [8, 12], "value": "requerimiento-hidrico", "confidence": 1 }, { "entity": "categoria", "location": [25, 32], "value": "siembra", "confidence": 1 }]
¿cuanta agua necesita mi cultivo?	**Entidad:** @categoria:requerimiento-hidrico **Intencion:** #describir	**Entidad:** @categoria:requerimiento-hidrico @categoria:siembra **Intencion:** #describir	intents: [{ "intent": "describir", "confidence": 0.9889055252075196 }], "entities": [{ "entity": "categoria", "location": [8, 12], "value": "requerimiento-hidrico", "confidence": 1 }, { "entity": "categoria", "location": [25, 32], "value": "siembra", "confidence": 1 }]
¿cuanta agua necesita el maiz?	Entidad: @categoria:requerimiento-hidrico @sujeto:maiz Intencion: #describir	Entidad: @categoria:requerimiento-hidrico @sujeto:maiz Intencion: #describir	intents: [{ "intent": "describir", "confidence": 1 }], "entities": [{ "entity": "categoria", "location": [8, 12], "value": "requerimiento-hidrico", "confidence": 1 }, { "entity": "sujeto", "location": [25, 29], "value": "maiz", "confidence": 1 }]
¿cuanto recurso hidrico necesita el maiz?	Entidad: @categoria:requerimiento-hidrico @sujeto:maiz Intencion: #describir	Entidad: @sujeto:maiz Intencion: #describir	intents: [{ "intent": "describir", "confidence": 0.5805310010910034 }], "entities": [{ "entity": "sujeto", "location": [36, 40], "value": "maiz", "confidence": 1 }]
¿que tanta agua necesita mi cultivo?	**Entidad:** @categoria:requerimiento-hidrico **Intencion:** #describir	Entidad: @categoria:requerimiento-hidrico @categoria:siembra Intencion: #describir	intents: [{ "intent": "describir", "confidence": 0.9879698276519775 }], "entities": [{ "entity": "categoria", "location": [11, 15], "value": "requerimiento-hidrico", "confidence": 1 }, { "entity": "categoria", "location": [28, 35], "value": "siembra", "confidence": 1 }]

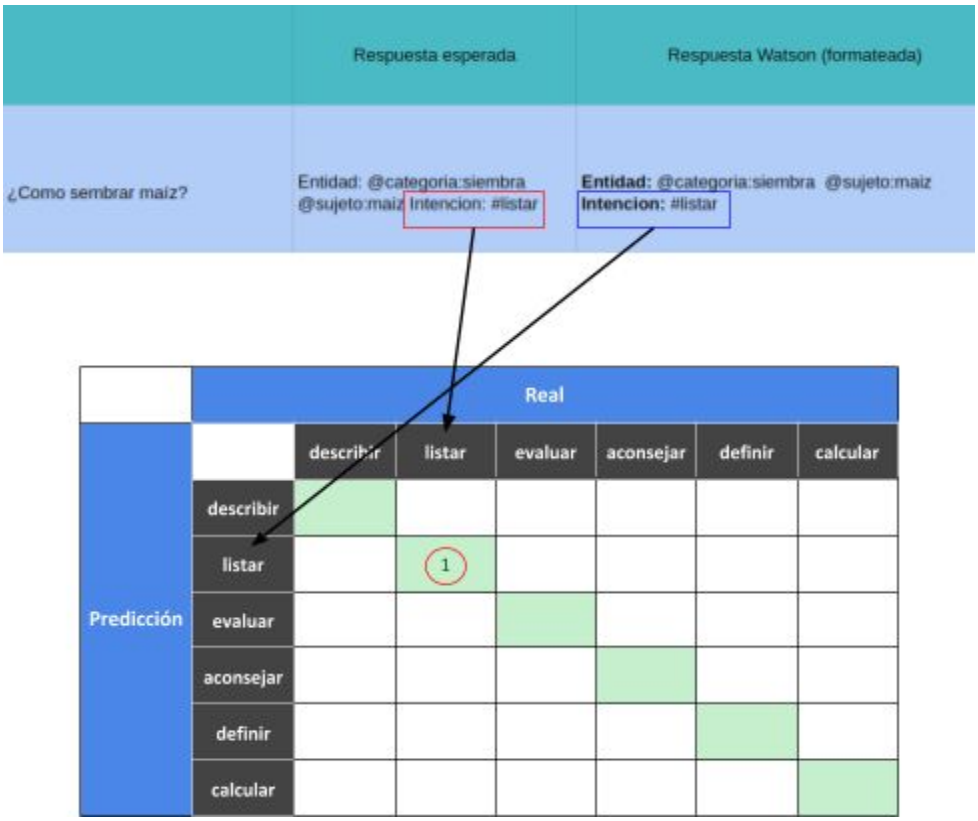
5.2 MEDIDAS DE DESEMPEÑO

Para obtener las medidas de desempeño y determinar qué tan correctamente funciona el modelo de clasificación de Watson Assistant, es necesario crear una matriz por cada variable categórica que se quiera evaluar. La matriz de confusión no es una medida de desempeño como tal, pero es necesaria para calcular las verdaderas medidas como: Exactitud, Precisión, *Recall* y F1.

En este caso se van a crear tres matrices de confusión: una para las intenciones, las entidades de tipo sujeto y otra para entidades de tipo categoría. Cada matriz de confusión va a contar con dos dimensiones: Real y Predicción, además de su conjunto de valores. Para llenar la matriz se toma la intención o entidad de las

columnas respuesta esperada y respuesta de Watson, luego se ubican en la matriz de confusión en la columna y fila correspondiente y se suma una unidad. Por ejemplo, para la matriz de confusión de intenciones el proceso anterior se representa en la Imagen 42. Ejemplo creación matriz de confusión.

Imagen 43. Ejemplo creación matriz de confusión



Este proceso de conteo se realiza para cada una de las 150 respuestas recibidas por los usuarios en las tres matrices de confusión. Las Matrices de confusión finales se pueden observar en las siguientes imágenes.

Imagen 44. Matriz de confusión de las intenciones

		Real					
		describir	listar	evaluar	aconsejar	definir	calcular
Prediccion	describir	90	5	0	2	2	0
	listar	15	3	0	1	0	0
	evaluar	0	0	2	0	0	0
	aconsejar	7	0	0	2	0	0
	definir	7	0	0	3	7	0
	calcular	1	0	0	0	2	1

Imagen 45. Matriz de confusión de las entidades “categoria”

		Real																
		siembra	fertilizacion-base	machas-foliares	requerimiento-hidrico	manejo-plagas	enfermedades	enfermedades-virus	calidad	nutricion	estado-vegetativo	estados-reproductivos	calidad-siembra	indice	recoleccion	antes-sembrar	estructura	vacio
Predicción	siembra	12	1	2	0	0	0	0	2	7	0	0	4	0	0	0	0	18
	fertilizacion-base	0	8		0	0	0	0	0	0	0	0	0	0	0	0	0	0
	machas-foliares	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	requerimiento-hidrico	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0
	manejo-plagas	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0
	enfermedades	0	0	1	0	0	0	2	0	0	0	0	0	0	0	0	0	4
	enfermedades-virus	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0
	calidad	0	0	0	0	0	0	0	2	0	0	0	2	0	0	0	0	0
	nutricion	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0
	estado-vegetativo	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0
	estados-reproductivos	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0
	calidad-siembra	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
	indice	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0
	recoleccion	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0
	antes-sembrar	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
	estructura	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	4	0
	Vacio	4	1	6	1	0	0	0	0	5	0	0	0	0	3	0	6	8

Imagen 46. Matriz de confusión de las entidades “sujeto”

		Real			
		maiz	semilla	fenologia	vacio
Prediccion	maiz	45	0	0	5
	semilla	0	21	0	0
	fenologia	0	0	10	0
	vacio	0	0	0	0

Luego de llenar todas las matrices de confusión, se llevó a cabo el cálculo de las medidas de Exactitud, Precisión, *Recall* y F1 para cada clase de las intenciones y las entidades. Como son demasiadas clases se decidió automatizar el cálculo de estas medidas, mediante el uso de un script de python que utiliza la librería sklearn. Los resultados con las medidas de desempeño para cada una de las matrices de confusión se presenta en las siguientes imágenes:

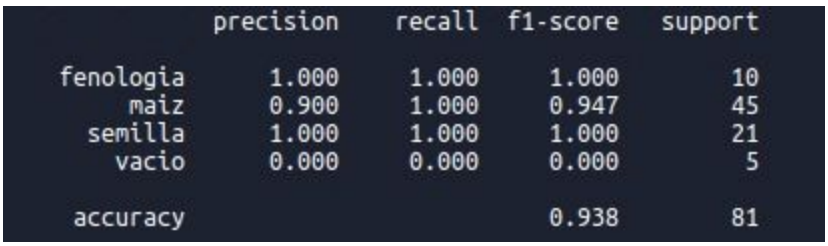
Imagen 47. Resumen de resultados para el clasificador de intenciones

	precision	recall	f1-score	support
Aconsejar	0.222	0.250	0.235	8
Calcular	0.250	1.000	0.400	1
Definir	0.412	0.636	0.500	11
Describir	0.909	0.750	0.822	120
Evaluar	1.000	1.000	1.000	2
Listar	0.158	0.375	0.222	8
accuracy			0.700	150
macro avg	0.492	0.669	0.530	150
weighted avg	0.793	0.700	0.735	150

Imagen 48. Resumen de resultados para el clasificador de entidades de tipo categoría

	precision	recall	f1-score	support
calidad	0.500	0.500	0.500	4
calidadSiembra	1.000	0.250	0.400	8
enfermedades	0.000	0.000	0.000	1
enfermedadesVirus	1.000	0.800	0.889	10
estadoReproductivo	1.000	1.000	1.000	10
estadoVegetativo	1.000	1.000	1.000	8
estructura	0.000	0.000	0.000	0
fertilizacion-base	1.000	0.800	0.889	10
indice	1.000	1.000	1.000	4
manchasFlorares	0.000	0.000	0.000	9
manejoPlagas	0.909	1.000	0.952	10
nutricion	1.000	0.409	0.581	22
recoleccion	1.000	0.750	0.857	12
requerimientoHidrico	1.000	0.909	0.952	11
siembra	0.500	1.000	0.667	16
vacio	0.000	0.000	0.000	0
accuracy			0.711	135
macro avg	0.682	0.589	0.605	135
weighted avg	0.845	0.711	0.731	135

Imagen 49. Resumen de resultados para el clasificador de entidades de tipo sujeto



	precision	recall	f1-score	support
fenologia	1.000	1.000	1.000	10
maiz	0.900	1.000	0.947	45
semilla	1.000	1.000	1.000	21
vacio	0.000	0.000	0.000	5
accuracy			0.938	81

Como se puede observar, las métricas aplicadas en la matriz de confusión de entidad de tipo sujeto, reflejan valores considerablemente altos. Esto se debe a que las entidades utilizadas tuvieron un entrenamiento correcto lo cual permitió que su identificación fuera acertada. Por otra parte, las entidades de tipo categoría como “manchas-foliares”, “antes-sembrar” y “calidad-siembra” sin los ejemplos de entrenamiento, es imposible de predecir por el modelo, es por eso que su exactitud es nula y esto baja considerablemente cualquier promedio de las medidas de todas las entidades. Se obtuvo que en promedio el modelo clasifica correctamente las entidades de tipo categoría con una precisión del 68% y un *recall* del 58% y las entidades de tipo sujeto tiene como *f1-score* el 93%.

Las medidas de desempeño para las intenciones reportaron valores bajos para intenciones como aconsejar,calcular y definir esto debido a la cantidad tan pequeña de ejemplos de entrenamiento y prueba que recibieron estas clases. Sin embargo para la intención describir que cuenta con el mayor número de ejemplos de entrenamiento, se obtuvo una precisión del 90% y un recall del 75% es decir de cada 10 preguntas el modelo clasifica correctamente 7 de ellas si su intención es describir. En promedio IBM Watson clasificó correctamente las intenciones con una precisión del 79% un recall de 70% y un f1-score de 73.5%.

Estos resultados son positivos para el proyecto y para la población de interés. La oportunidad tecnológica de ayudar a la población rural desde una llamada telefónica para aumentar el interés y la participación, en los estudios de áreas del conocimiento relacionadas al campo, los convierte en una población con competencias laborales más valoradas.

Con el fin de tener una mejora continua dentro del proyecto, se decidió guardar las preguntas que cada usuario realiza durante la llamada, para que luego sean revisadas por un experto y si es el caso, agregarlas e implementarlas dentro del

dataset de entrenamiento de Watson Assistant. Así a medida que sea usado aprenderá cada vez más con el tiempo.

6. CONCLUSIONES

En el presente trabajo, se logró la construcción de un método para implementar el procesamiento de lenguaje natural dentro de un *voicebot*, mediante la integración de diferentes y herramientas y aplicaciones en el mercado, con el fin de mejorar su capacidad para responder preguntas abiertas de un usuario, sobre el cultivo de maíz. En la implementación del método propuesto, se utilizó IBM Watson Assistant, como herramienta de PNL por los excelentes resultados para clasificar intenciones al igual que por sus características únicas dentro de la plataforma, como el módulo de entrenamiento y testeo gráfico.

De acuerdo con lo señalado y los resultados demostrados, Watson Assistant puede ser entrenado con un dataset de información pequeño y obtener medidas de desempeño como la precisión y exactitud bastante altas. Sin embargo, por la falta de balanceo en los datos de prueba, se puede presentar un sesgo en los resultados. En nuestro caso la intención que presenta este sesgo es “describir”, puesto que, en el experimento de prueba, fue la intención que más tuvo ejemplos de pregunta, por ende, el modelo solo clasifica correctamente con una precisión del 90%, si recibe esa intención como entrada.

La implementación del diseño de una ontología, permitió reutilizar la misma estructura de información entre diferentes aplicaciones, lo que permitió un acoplamiento entre la herramienta NLP y la base de datos. La construcción de una API permitió la integración de funcionalidades de diferentes aplicaciones de una forma sencilla, que permite la iteración y continua mejora del método propuesto.

Para garantizar mejores resultados a largo plazo en necesario un *dataset* de entrenamiento más grande de información, así como un *dataset* de pruebas completamente balanceado, para poder garantizar que los resultados obtenidos se aplican en todo el modelo y no solo en intenciones o entidades específicas.

7. TRABAJOS FUTUROS

Durante el desarrollo de este proyecto han surgido muchos caminos abiertos en donde se puede seguir trabajando y mejorar, algunos relacionados directamente al presente trabajo y otros son posibles nuevas líneas de investigación que por el alcance de este proyecto no pueden ser tenidos en cuenta, a continuación se muestra una lista de trabajos futuros, desarrollos específicos y propuestas para que el proyecto tenga más apoyo y pueda ser mejorado, con el fin de tener un resultado óptimo y un mayor impacto:.

- Plantear el uso de una herramienta de NLU diferente, de la misma manera que utilizar otro tipo de guía o metodología para la realización de la ontología
- Implementar un ciclo de vida de software en el desarrollo del proyecto
- Agregar más fuentes de información que permitan al voicebot responder preguntas de temas diferentes a los tratados en el audiolibro
- Extender la ontología para que soporte otros idiomas y el voicebot pueda llegar a otros países, rompiendo brechas de idioma impactando nuevas comunidades.
- Diseño, desarrollo e implementación de una nueva infraestructura para el manejo de información, utilizando un lenguaje diferente y un nuevo motor de búsqueda
- Realizar un sondeo previo sobre conceptos del maíz con personas expertas para replantear las entidades, intenciones y las preguntas de entrenamiento utilizadas
- Implementar el modelo por medio de la herramienta de NLU en un software de terceros para que pueda ser utilizado en plataformas tecnológicas y así llegar a un público mayor.

BIBLIOGRAFÍA

Acerca de JavaScript. *Documentación web de MDN* [en línea] [Consulta: 27 junio 2019]. Disponible en: https://developer.mozilla.org/es/docs/Web/JavaScript/Acerca_de_JavaScript.

ALTINOK, D. 2019. An Ontology-Based Dialogue Management System for Banking and Finance Dialogue Systems. arXiv.org [en línea]. [Consulta: 15 julio 2019]. Disponible en: https://arxiv.org/abs/1804.04838.

AZETA, A., AYO, C. and OMOREGBE, N. 2019. A Voice-Enabled framework for recommender and adaptation systems in E-learning. [en línea]. [Consulta: 2 agosto 2019]. Disponible en: https://www.researchgate.net/publication/286607659_A_Voice-Enabled_framework_for_recommender_and_adaptation_systems_in_E-learning.

CALIXTO, A. and MAGALHÃES, V. 2019. MediBot: An Ontology based Chatbot for Portuguese Speakers Drug's Users. [en línea]. [Consulta: 5 agosto 2019]. Disponible en: https://www.researchgate.net/publication/333432674_MediBot_An_Ontology_based_Chatbot_for_Portuguese_Speakers_Drug's_Users.

CLARIZIA1, F., COLACE1, F., LOMBARDI1, M., PASCALE1, F. and SANTANIELLO2, D. 2019. Chatbot: An Education Support System for Student. *Chatbot: An Education Support System for Student* [en línea]. [Consulta: 12 julio 2019]. Disponible en: https://link.springer.com/chapter/10.1007/978-3-030-01689-0_23.

CHAND, S. 2019. What Is Elasticsearch - Getting Started | ELK Stack | Edureka. *Edureka*[en línea]. [Consulta: 19 junio 2019]. Disponible en: https://www.edureka.co/blog/what-is-elasticsearch/.

COLOMBIA, MINISTERIO DE LAS TIC 2019. Internet satelital, opción para conectar el campo y mejorar la competitividad - Ministerio de Tecnologías de la Información y las Comunicaciones. Mintic.gov.co [en línea]. [Consulta: 9 julio 2019]. Disponible en: https://www.mintic.gov.co/portal/604/w3-article-100374.html.

COLLINA, M. and COPES, F. 2019. Introduction to Node.js. *Introduction to Node.js* [en línea]. [Consulta: 21 junio 2019]. Disponible en: https://nodejs.dev/.

DIGIUM 2019. Get Started. Asterisk.org [en línea]. [Consulta: 18 octubre 2019]. Disponible en: https://www.asterisk.org/get-started.

ELASTIC 2019. Elasticsearch: RESTful, Distributed Search & Analytics | Elastic. *Elastic.co*[en línea]. [Consulta: 5 julio 2019]. Disponible en: https://www.elastic.co/products/elasticsearch.

ESCOBAR, G. 2019. Git y Github. El Blog de Make it Real [en línea]. [Consulta: 20 octubre 2019]. Disponible en: https://blog.makeitreal.camp/git-y-github/.

ESCOBAR, G. 2019. La relevancia de la agricultura en América Latina y el Caribe. Nuso.org [en línea]. [Consulta: 12 julio 2019]. Disponible en: https://nuso.org/media/documents/agricultura.pdf.

F. NOY, N. and L. MCGUINNESS, D. 2019. Ontology Development 101: A Guide to Creating Your First Ontology [en línea]. Stanford: s.n. [Consulta: 19 septiembre 2019]. Disponible en: https://protege.stanford.edu/publications/ontology_development/ontology101.pdf.

GONZÁLEZ, A. 2019. ¿Qué es Machine Learning? – Cleverdata. Cleverdata.io [en línea]. [Consulta: 18 noviembre 2019]. Disponible en: https://cleverdata.io/que-es-machine-learning-big-data/.

GUERRERO MONTAÑA, M. 2019. MODELO PREDICTIVO PARA INFERIR EL PRÓXIMO PRESIDENTE DE ESTADO MEDIANTE EL USO DE ONTOLOGÍAS EN TWITTER. Pre-grado. S.l.: Universidad Católica De Colombia.

GUZMÁN LUNA, J., LÓPEZ BONILLA, M. and TORRES, I. 2019. Metodologías y métodos para la construcción de ontologías [en línea]. Medellín: s.n. [Consulta: 14 octubre 2019]. Disponible en: https://revistas.utp.edu.co/index.php/revistaciencia/article/view/6693.

HASAN, I. 2019. A Practical Introduction to Elasticsearch | Elastic Blog. *Elastic Blog* [en línea]. [Consulta: 6 julio 2019]. Disponible en: https://www.elastic.co/blog/a-practical-introduction-to-elasticsearch.

HAYES, B. 2019. Using IBM Watson to Answer Two Important Questions about your Customers |. *Businessoverbroadway.com* [en línea]. [Consulta: 28 julio 2019]. Disponible en: http://businessoverbroadway.com/2019/03/13/using-ibm-watson-to-answer-two-important-questions-about-your-customers/.

HOPE, C. 2019. What is Voice Recognition?. *Computerhope.com* [en línea]. [Consulta: 29 junio 2019]. Disponible en: https://www.computerhope.com/jargon/v/voicreco.htm.

IBAÑEZ, A. 2019. Tendencias globales para el mercado de los asistentes virtuales. Nae.global [en línea]. [Consulta: 17 julio 2019]. Disponible en: https://nae.global/tendencias-globales-para-el-mercado-de-los-asistentes-virtuales/

IONOS 2019. Qué es Elasticsearch. IONOS Digitalguide [en línea]. [Consulta: 11 noviembre 2019]. Disponible en: https://www.ionos.es/digitalguide/servidores/configuracion/que-es-elasticsearch/.

JASMIN RANI, P., BAKTHAKUMAR, J., KUMAAR, B., KUMAAR, U. and KUMAR, S. 2019. Voice controlled home automation system using Natural Language Processing (NLP) and Internet of Things (IoT) - IEEE Conference Publication. *Ieeexplore.ieee.org* [en línea]. [Consulta: 10 agosto 2019]. Disponible en: https://ieeexplore.ieee.org/document/8261311.

JavaScript. *Documentación web de MDN* [en línea] [Consulta: 27 junio 2019]. Disponible en: https://developer.mozilla.org/es/docs/Web/JavaScript.

KHATTER, K., KOLI'S, A., KHURANA'S, D. and SINGH, S. 2019. Voice controlled home automation system using Natural Language Processing (NLP) and Internet of Things (IoT). [en línea]. [Consulta: 10 agosto 2019]. Disponible en: https://www.researchgate.net/publication/319164243_Natural_Language_Processing_State_of_The_Art_Current_Trends_and_Challenges.

KINSTA 2019. ¿Qué es GitHub? Una Guía para Principiantes sobre GitHub. Kinsta WordPress Hosting Gestionado [en línea]. [Consulta: 25 octubre 2019]. Disponible en: https://kinsta.com/es/base-de-conocimiento/que-es-github/.

LOMBARDI, M., COLACE, F. and PASCALE, F. 2019. Chatbot: An Education Support System for Student: 10th International Symposium, CSS 2018, Amalfi, Italy, October 29–31, 2018, Proceedings. [en línea]. [Consulta: 31 julio 2019]. Disponible en: https://www.researchgate.net/publication/327831707_Chatbot_An_Education_Support_System_for_Student_10th_International_Symposium_CSS_2018_Amalfi_Italy_October_29-31_2018_Proceedings.

MDN CONTRIBUTORS 2019. Express/Node introduction. MDN Web Docs [en línea]. [Consulta: 19 octubre 2019]. Disponible en: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction.

MERCADO, B. 2019. Análisis: ¿cómo superar la crisis de educación en el campo colombiano?. Semana [en línea]. [Consulta: 12 noviembre 2019]. Disponible en:

http://semana.com/nacion/articulo/panorama-de-la-educacion-en-el-campo-colombiano/531885.

MICROSOFT 2019. Documentation for Visual Studio Code. Code.visualstudio.com [en línea]. [Consulta: 27 octubre 2019]. Disponible en: https://code.visualstudio.com/docs.

MURADAS, Y. 2019. Qué es NPM y para qué sirve. OpenWebinars.net [en línea]. [Consulta: 24 octubre 2019]. Disponible en: https://openwebinars.net/blog/que-es-node-package-manager/.

NÚÑEZ, J. 2019. » Google Forms: Una Herramienta que nos ayudará con las Encuestas. Modalidad A Distancia [en línea]. [Consulta: 9 noviembre 2019]. Disponible en: http://blog.continental.edu.pe/uc-virtual/una-herramienta-que-nos-ayudara-con-las-encuestas/.

PAPACCHINI, F. 2019. *Proteg´ e Tutorial* [en línea]. Liverpool: s.n. [Consulta: 30 junio 2019]. Disponible en: http://cgi.csc.liv.ac.uk/~frank/teaching/comp08/protege_tutorial.pdf.

PERDOMO B., A. 2019. MilAgro por la educación tecnológica en el campo colombiano - ANEIA - Universidad de Los Andes. ANEIA - Universidad de Los Andes [en línea]. [Consulta: 10 agosto 2019]. Disponible en: https://agronegocios.uniandes.edu.co/2015/09/17/milagro-por-la-educacion-tecnologica-en-el-campo-colombiano/.

RESEARCH, S. 2019. protégé. Protege.stanford.edu [en línea]. [Consulta: 29 julio 2019]. Disponible en: https://protege.stanford.edu/products.php#web-protege.

ROUSE, M. 2019. What is ElasticSearch? - Definition from WhatIs.com. *WhatIs.com* [en línea]. [Consulta: 21 junio 2019]. Disponible en: https://whatis.techtarget.com/definition/ElasticSearch.

ROUSE, M. 2019. What is IBM Watson supercomputer? - Definition from WhatIs.com. *SearchEnterpriseAI* [en línea]. [Consulta: 24 junio 2019]. Disponible en: https://searchenterpriseai.techtarget.com/definition/IBM-Watson-supercomputer.

ROUSE, M. 2019. What is natural language? - Definition from WhatIs.com. *WhatIs.com* [en línea]. [Consulta: 26 junio 2019]. Disponible en: https://whatis.techtarget.com/definition/natural-language.

ROUSE, M. 2019. What is Natural language processing (NLP)? A definition from WhatIs.com. *SearchBusinessAnalytics* [en línea]. [Consulta: 28 junio 2019]. Disponible en: https://searchbusinessanalytics.techtarget.com/definition/natural-language-processing-NLP.

ROUSE, M. 2019. What is natural language understanding (NLU)? - Definition from WhatIs.com. *SearchEnterpriseAI* [en línea]. [Consulta: 30 junio 2019]. Disponible en: https://searchenterpriseai.techtarget.com/definition/natural-language-understanding-NLU.

ROUSE, M. 2019. What is ontology? - Definition from WhatIs.com. *WhatIs.com* [en línea]. [Consulta: 25 junio 2019]. Disponible en: https://whatis.techtarget.com/definition/ontology.

SAS 2019. ¿Qué es deep learning?. Sas.com [en línea]. [Consulta: 13 noviembre 2019]. Disponible en: https://www.sas.com/es_ar/insights/analytics/deep-learning.html.

SITIO BIG DATA 2019. Machine Learning: Seleccion Métricas de clasificación - sitiobigdata.com. sitiobigdata.com [en línea]. [Consulta: 16 noviembre 2019]. Disponible en: http://sitiobigdata.com/2019/01/19/machine-learning-metrica-clasificacion-parte-3/.

STANFORD UNIVERSITY 2019. protégé. Protege.stanford.edu [en línea]. [Consulta: 15 septiembre 2019]. Disponible en: https://protege.stanford.edu/.

Terminal. Sistemas.com [en línea] [Consulta: 24 octubre 2019]. Disponible en: https://sistemas.com/terminal.php.

VEGESNA, A., JAIN, P. and PORWAL, D. 2019. *Ontology based Chatbot (For E-commerce Website)* [en línea]. S.l.: s.n. [Consulta: 6 agosto 2019]. Disponible en: https://pdfs.semanticscholar.org/a8a5/8009b1afc0dfc68a61e5a917e290c7c9579e.pdf.

Watson Natural Language Understanding - Overview - Philippines. *Ibm.com* [en línea]. [Consulta: 29 junio 2019]. Disponible en: https://www.ibm.com/ph-en/marketplace/natural-language-understanding.

What is Node.js?. *Tutorialsteacher.com* [en línea] [Consulta: 24 junio 2019]. Disponible en: https://www.tutorialsteacher.com/nodejs/what-is-nodejs.